





DATA STRUCTURES LAB

File

COURSE CODE: ARD 255

***2022-2023***

**Submitted By:**

**Name: Shivam Kumar Jha**

**Enroll No.: 06119011721**

**Branch: IIOT (Batch 2)**

**Submitted To:**

**Dr. Neetu Mahajan (Assistant Professor)**

**Practical - 1**

**Q1.Write a program to implement addition and multiplication of two 2D array.**

Code:

#include <stdlib.h>

#include <stdio.h>

int main()

{

int row, col, k, a[10][10], b[10][10], sum[10][10], result[10][10], i, j;

printf("Enter the number of rows (between 1 and 10): ");

scanf("%d", &row);

printf("Enter the number of columns (between 1 and 10): ");

scanf("%d", &col);

printf("\nEnter elements of 1st matrix:\n");

for (i = 0; i < row; ++i)

for (j = 0; j < col; ++j)

{

printf("Enter element a%d%d: ", i + 1, j + 1);

scanf("%d", &a[i][j]);

}

printf("Enter elements of 2nd matrix:\n");

for (i = 0; i < row; ++i)

for (j = 0; j < col; ++j)

{

printf("Enter element b%d%d: ", i + 1, j + 1);

scanf("%d", &b[i][j]);

}

// adding two matrices

for (i = 0; i < row; ++i)

for (j = 0; j < col; ++j)

{

sum[i][j] = a[i][j] + b[i][j];

}

printf("Sum of the two matrices: \n");

for (i = 0; i < row; ++i){

for (j = 0; j < col; ++j)

{

printf("%d ", sum[i][j]);

}

printf("\n\n");

}

// multiplication of two matrix

for (int i = 0; i < row; ++i)

{

for (int j = 0; j < col; ++j)

{

for (int k = 0; k < col; ++k)

{

result[i][j] = a[i][k] \* b[k][j];

}

}

}

printf("\nOutput Matrix:\n");

for (int i = 0; i < row; ++i)

{

for (int j = 0; j < col; ++j)

{

printf("%d ", result[i][j]);

if (j == col - 1)

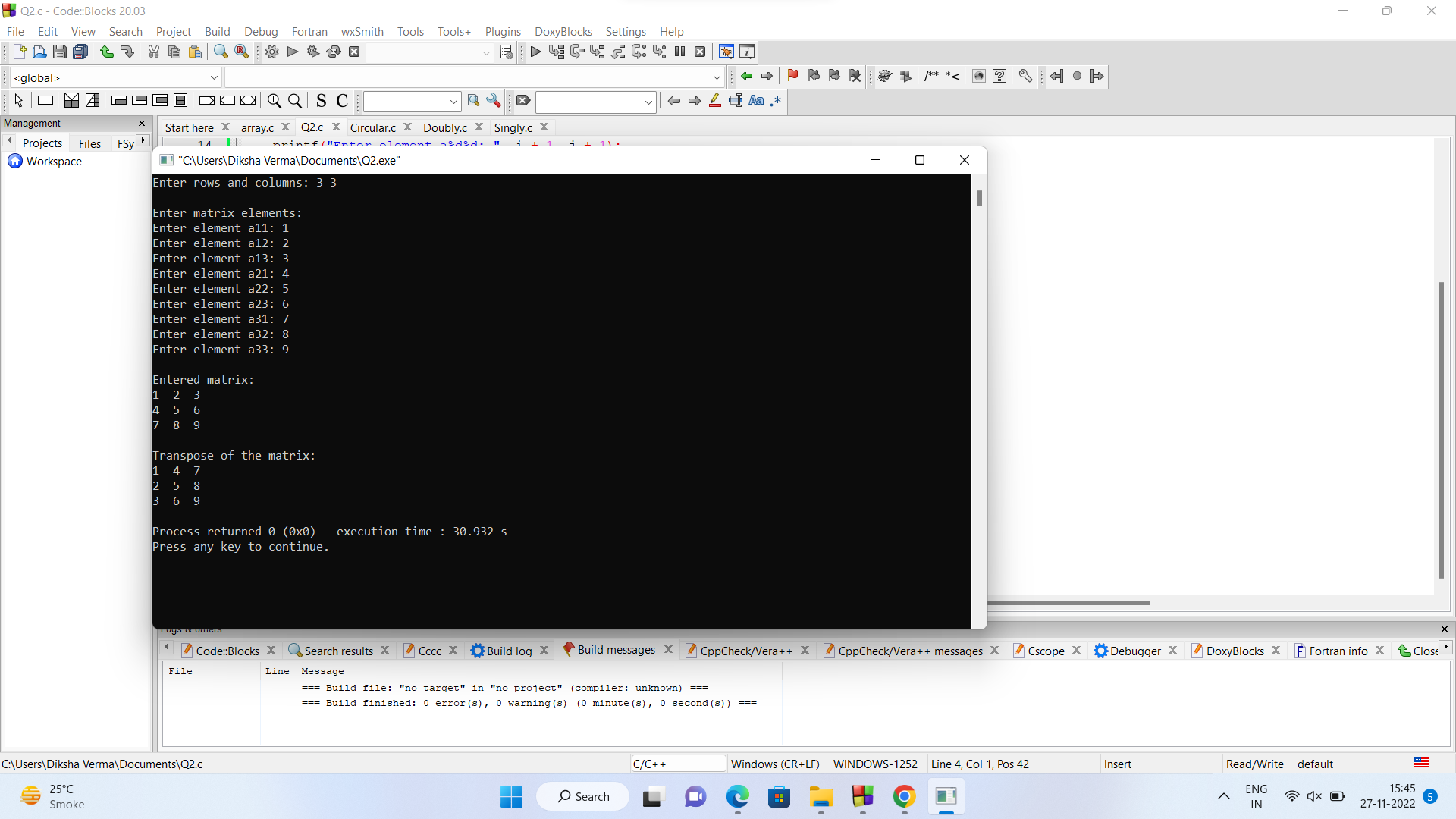
printf("\n");

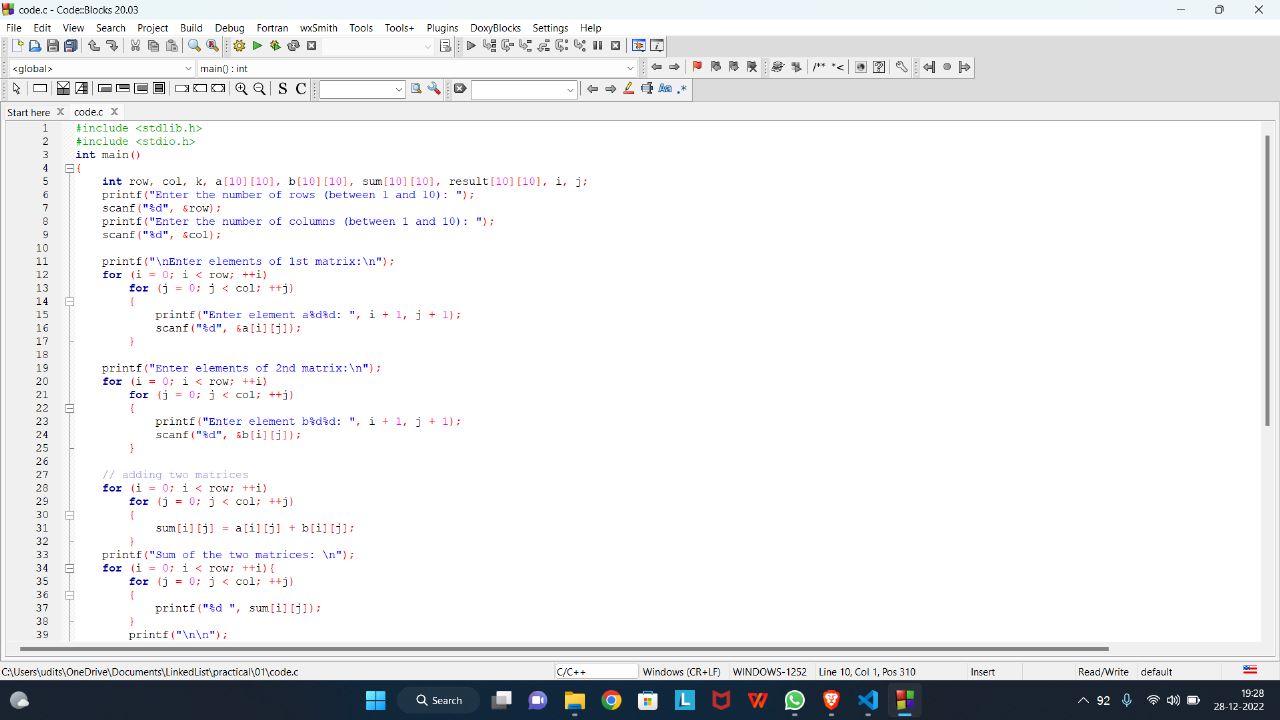
}

}

return 0;

}

Code Screeshot 



**Practical - 2**

**Q2.Write a program to transpose a 2D array.**

Code:

#include<stdlib.h>

#include <stdio.h>

int main() {

int a[10][10], transpose[10][10], r, c;

printf("Enter rows and columns: ");

scanf("%d %d", &r, &c);

// asssigning elements to the matrix

printf("\nEnter matrix elements:\n");

for (int i = 0; i < r; ++i)

for (int j = 0; j < c; ++j) {

printf("Enter element a%d%d: ", i + 1, j + 1);

scanf("%d", &a[i][j]);

}

// printing the matrix a[][]

printf("\nEntered matrix: \n");

for (int i = 0; i < r; ++i)

for (int j = 0; j < c; ++j) {

printf("%d ", a[i][j]);

if (j == c - 1)

printf("\n");

}

// computing the transpose

for (int i = 0; i < r; ++i)

for (int j = 0; j < c; ++j) {

transpose[j][i] = a[i][j];

}

// printing the transpose

printf("\nTranspose of the matrix:\n");

for (int i = 0; i < c; ++i)

for (int j = 0; j < r; ++j) {

printf("%d ", transpose[i][j]);

if (j == r - 1)

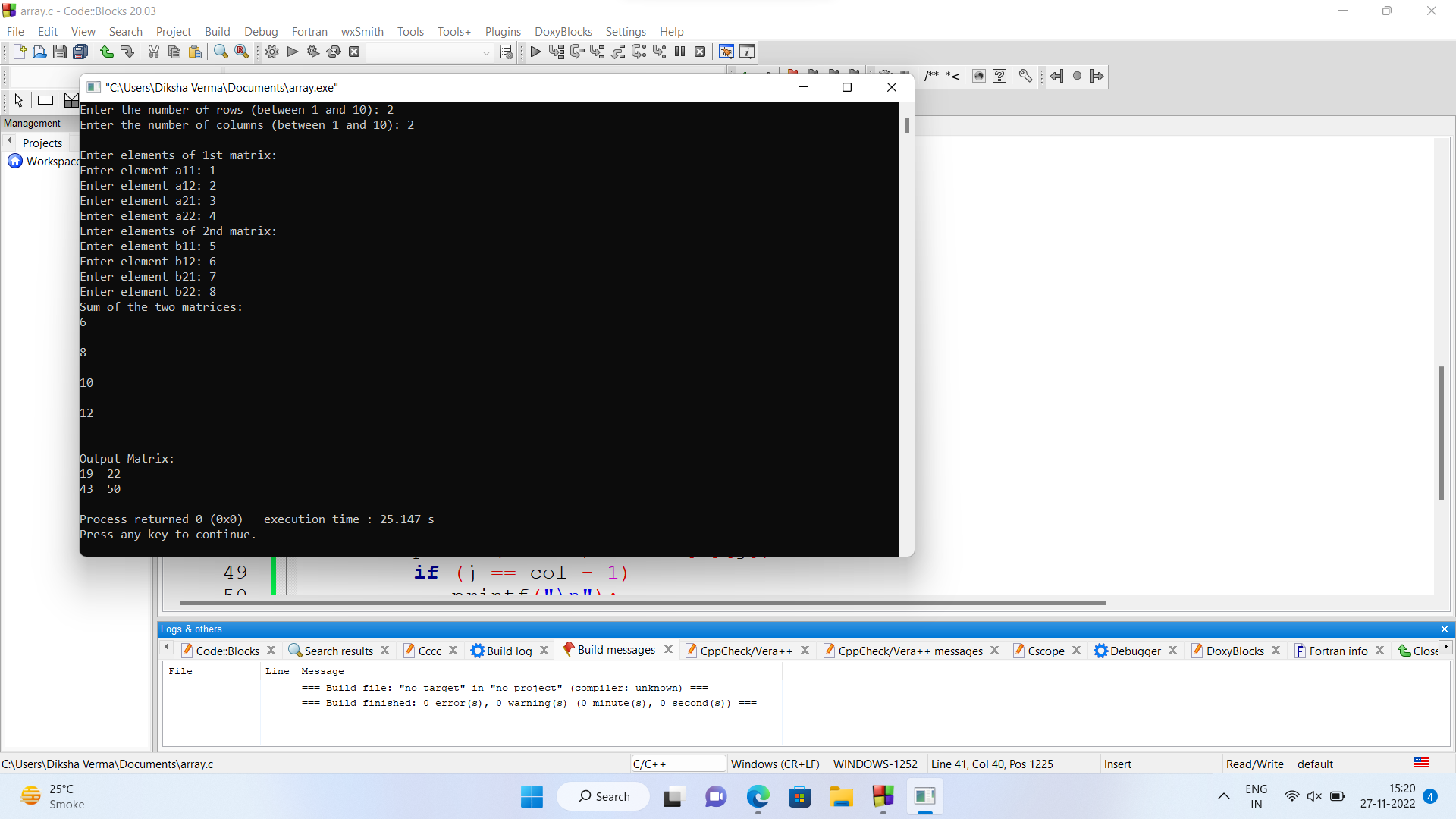
printf("\n");

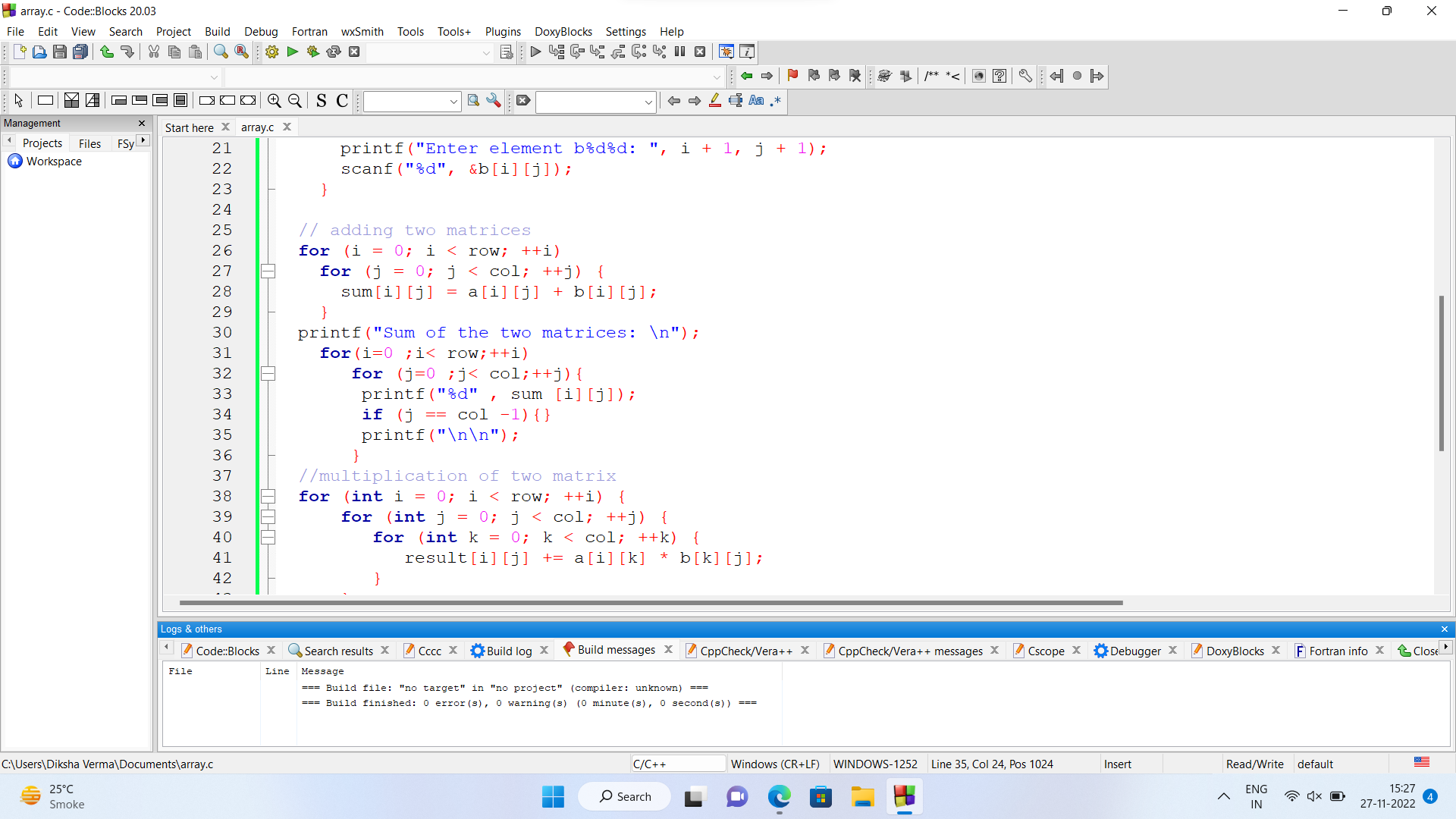
}

return 0;

}

Code Screenshot:





**Practical - 3**

**Q3.Write a program to implement**

1. **Singly Link List**

Code:

#include<stdlib.h>

#include <stdio.h>

int main() {

int a[10][10], transpose[10][10], r, c;

printf("Enter rows and columns: ");

scanf("%d %d", &r, &c);

// asssigning elements to the matrix

printf("\nEnter matrix elements:\n");

for (int i = 0; i < r; ++i)

for (int j = 0; j < c; ++j) {

printf("Enter element a%d%d: ", i + 1, j + 1);

scanf("%d", &a[i][j]);

}

// printing the matrix a[][]

printf("\nEntered matrix: \n");

for (int i = 0; i < r; ++i)

for (int j = 0; j < c; ++j) {

printf("%d ", a[i][j]);

if (j == c - 1)

printf("\n");

}

// computing the transpose

for (int i = 0; i < r; ++i)

for (int j = 0; j < c; ++j) {

transpose[j][i] = a[i][j];

}

// printing the transpose

printf("\nTranspose of the matrix:\n");

for (int i = 0; i < c; ++i)

for (int j = 0; j < r; ++j) {

printf("%d ", transpose[i][j]);

if (j == r - 1)

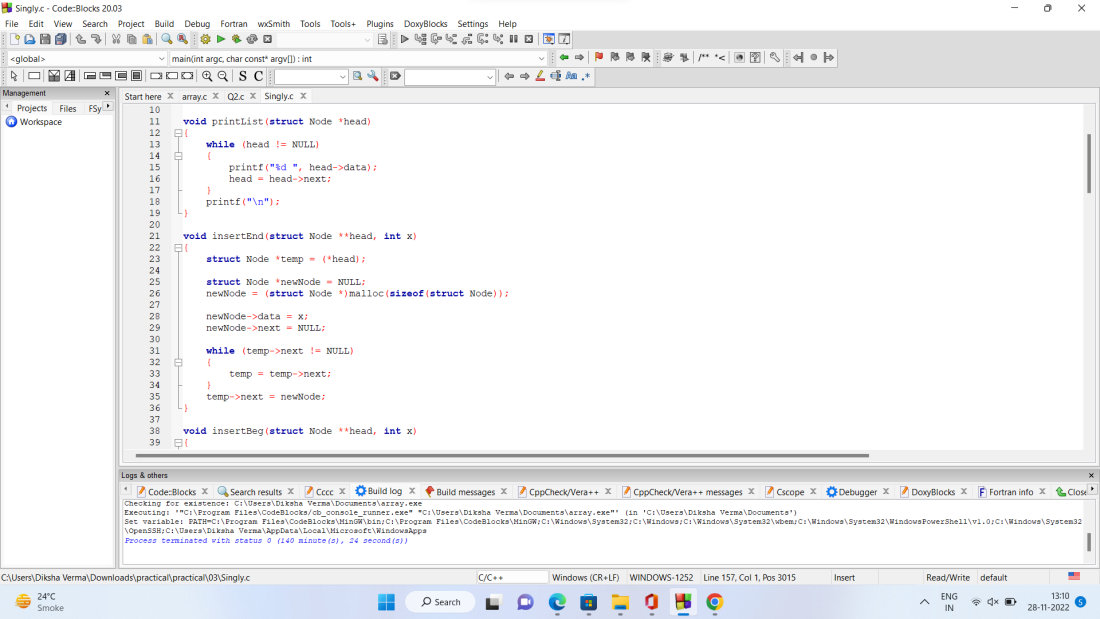
printf("\n");

}

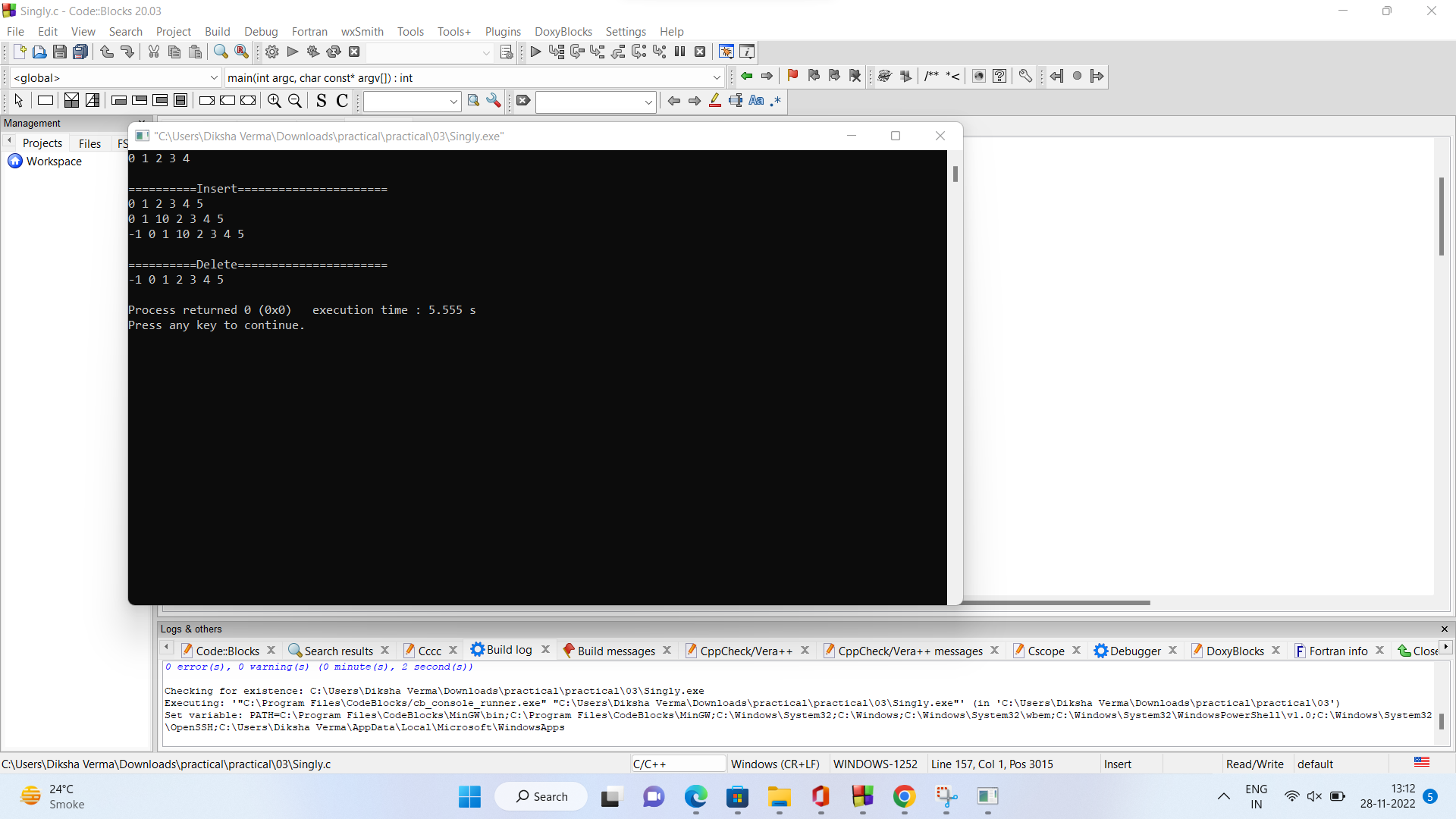
return 0;

}

Code Screenshot:



Output Screenshot:



B. **Doubly Link List**

Code:

#include <stdlib.h>

#include <stdio.h>

//============================Doubly==========================//

struct Node

{

int data;

struct Node \*next;

struct Node \*prev;

};

void insertBeg(struct Node \*\*head, int x)

{

struct Node \*curr = (\*head);

struct Node \*newNode = NULL;

newNode = (struct Node \*)malloc(sizeof(struct Node));

newNode->data = x;

newNode->prev = NULL;

newNode->next = (\*head);

// newNode->next=curr;

(\*head)->prev = newNode;

// curr->prev=newNode;

(\*head) = newNode;

}

void insertEnd(struct Node \*\*head, int x)

{

struct Node \*curr = (\*head);

struct Node \*newNode = NULL;

newNode = (struct Node \*)malloc(sizeof(struct Node));

while (curr->next != NULL)

{

curr = curr->next;

}

curr->next = newNode;

newNode->prev = curr;

newNode->data = x;

newNode->next = NULL;

}

void insertMid(struct Node \*\*head, int x, int pos)

{

struct Node \*curr = (\*head);

struct Node \*newNode = NULL;

newNode = (struct Node \*)malloc(sizeof(struct Node));

for (int i = 1; i < pos; i++)

{

curr = curr->next;

}

newNode->data = x;

newNode->next = curr->next;

newNode->prev = curr;

curr->next->prev = newNode;

curr->next = newNode;

}

void deleteBig(struct Node \*\*head)

{

struct Node \*curr = (\*head)->next;

struct Node \*temp = (\*head);

curr->prev = NULL;

temp->next = NULL;

(\*head) = curr;

free(temp);

}

void deleteEnd(struct Node \*\*head)

{

struct Node \*curr = (\*head);

while (curr->next->next != NULL)

{

curr = curr->next;

}

struct Node \*temp = curr->next;

curr->next = NULL;

temp->prev = NULL;

free(temp);

}

void deleteMid(struct Node \*\*head, int pos)

{

struct Node \*curr = (\*head);

for (int i = 1; i < pos - 1; i++)

{

curr = curr->next;

}

struct Node \*temp = curr->next;

temp->next->prev = curr;

curr->next = curr->next->next;

free(temp);

}

void printList(struct Node \*head)

{

while (head != NULL)

{

printf("%d ", head->data);

head = head->next;

}

printf("\n");

}

int main(int argc, char const \*argv[])

{

struct Node \*head = NULL;

struct Node \*one = NULL;

struct Node \*two = NULL;

struct Node \*three = NULL;

struct Node \*four = NULL;

head = (struct Node \*)malloc(sizeof(struct Node));

one = (struct Node \*)malloc(sizeof(struct Node));

two = (struct Node \*)malloc(sizeof(struct Node));

three = (struct Node \*)malloc(sizeof(struct Node));

four = (struct Node \*)malloc(sizeof(struct Node));

head->data = 0;

head->next = one;

head->prev = NULL;

one->data = 1;

one->next = two;

one->prev = head;

two->data = 2;

two->next = three;

two->prev = one;

three->data = 3;

three->next = four;

three->prev = two;

four->data = 4;

four->next = NULL;

four->prev = three;

printList(head);

printf("\n==========Insert======================\n");

insertBeg(&head, -1);

printList(head);

insertMid(&head, 20, 2);

printList(head);

insertEnd(&head, 5);

printList(head);

printf("\n==========Delete======================\n");

deleteBig(&head);

printList(head);

deleteMid(&head,3);

printList(head);

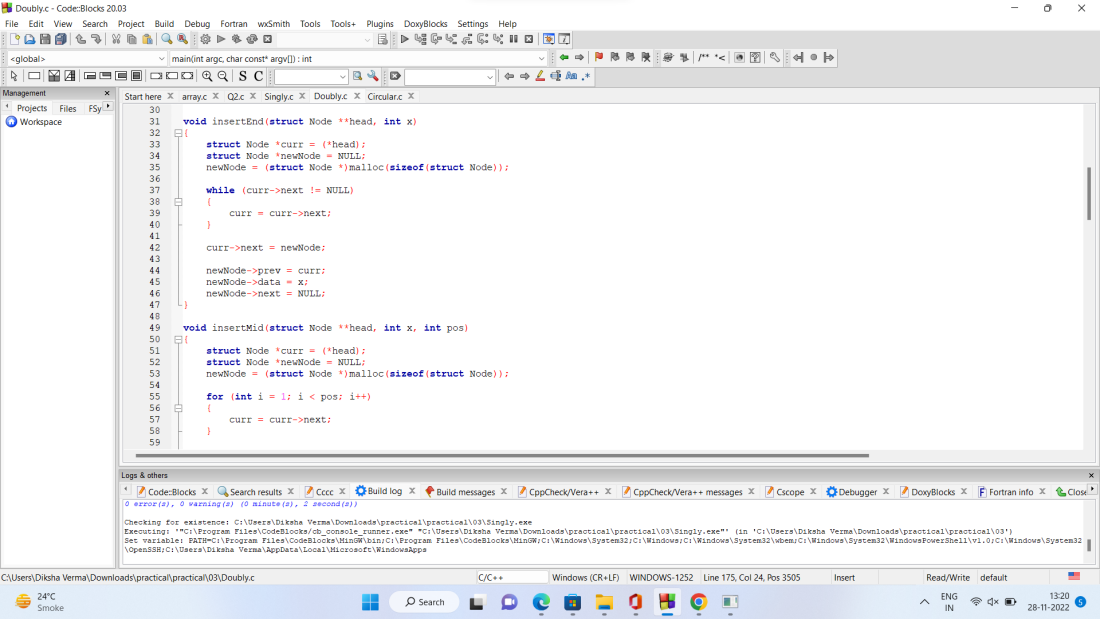
deleteEnd(&head);

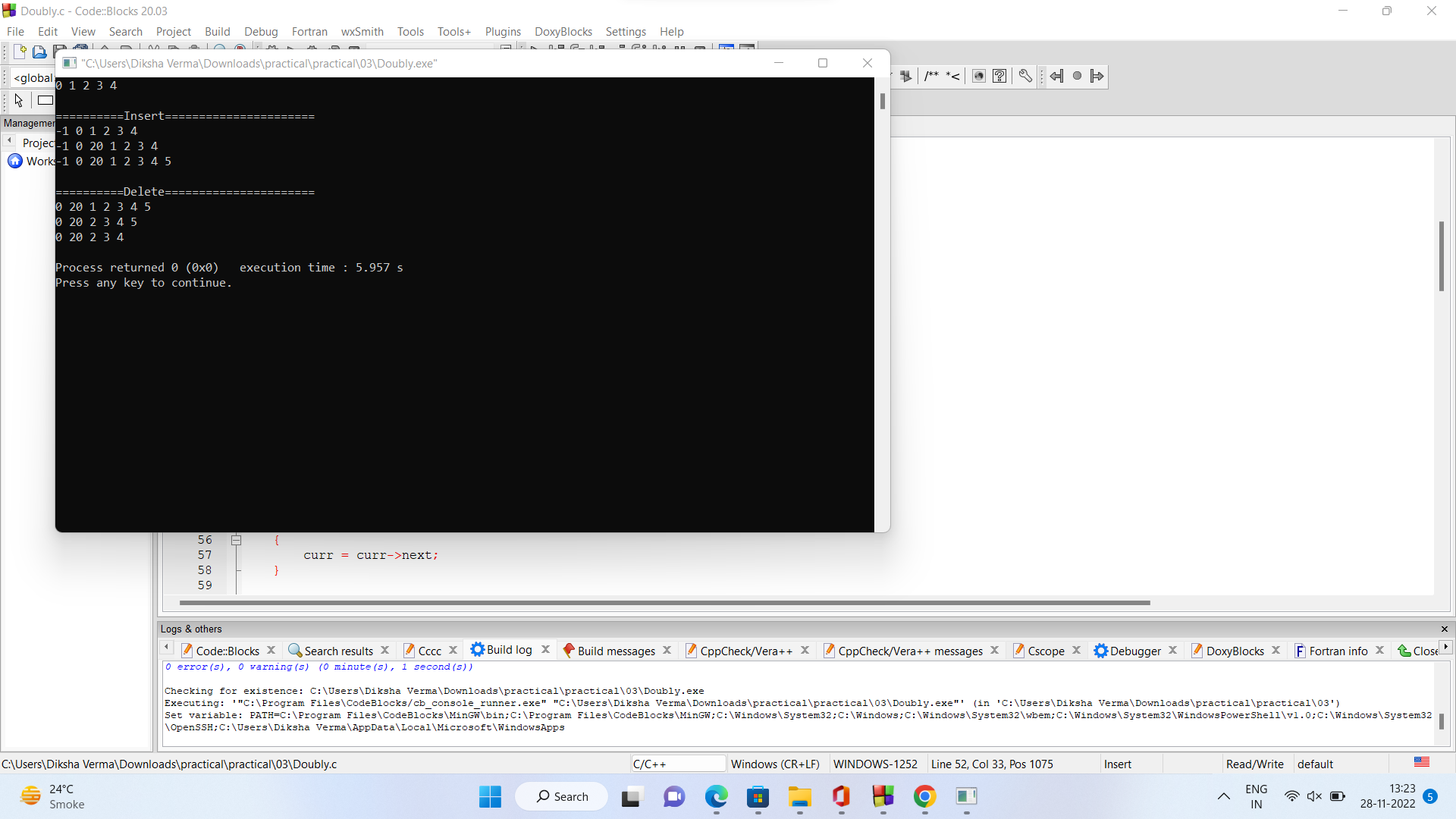
printList(head);

return 0;

}

Code Screenshot:



Output Screenshot: 

c. **Circular Link list**

Code:

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

// ==============================circularly Singly ====================//

struct Node

{

int data;

struct Node \*next;

};

void insertEnd(struct Node \*\*head, int x)

{

struct Node \*curr = (\*head);

struct Node \*newNode = NULL;

newNode = (struct Node \*)malloc(sizeof(struct Node));

do

{

curr = curr->next;

} while (curr->next != (\*head));

newNode->data = x;

newNode->next = curr->next;

curr->next = newNode;

}

void insertBeg(struct Node \*\*head, int x)

{

struct Node \*curr = \*(head);

struct Node \*newNode = NULL;

newNode = (struct Node \*)malloc(sizeof(struct Node));

do

{

curr = curr->next;

} while (curr->next != (\*head));

newNode->data = x;

newNode->next = (\*head);

curr->next = newNode;

(\*head) = newNode;

}

void insertMid(struct Node \*\*head, int x, int pos)

{

struct Node \*curr = (\*head);

struct Node \*newNode = NULL;

newNode = (struct Node \*)malloc(sizeof(struct Node));

for (int i = 1; i < pos; i++)

{

curr = curr->next;

}

newNode->data = x;

newNode->next = curr->next;

curr->next = newNode;

}

void deleteBeg(struct Node \*\*head)

{

struct Node \*curr = (\*head);

do

{

curr = curr->next;

} while (curr->next != (\*head));

struct Node \*temp = curr->next;

curr->next = curr->next->next;

free(temp);

(\*head) = curr->next;

}

void deleteEnd(struct Node \*\*head)

{

struct Node \*curr = (\*head);

do

{

curr = curr->next;

} while (curr->next->next != (\*head));

struct Node \*tail = curr->next;

curr->next = curr->next->next;

free(tail);

}

void deleteMid(struct Node \*\*head, int pos)

{

struct Node \*curr = (\*head);

for (int i = 1; i < pos - 1; i++)

{

curr = curr->next;

}

struct Node \*temp = curr->next;

curr->next = temp->next;

free(temp);

}

void printList(struct Node \*\*head)

{

struct Node \*curr = (\*head);

do

{

printf("%d ", curr->data);

curr = curr->next;

} while (curr != (\*head));

printf("\n");

}

int main(int argc, char const \*argv[])

{

struct Node \*head = NULL;

struct Node \*one = NULL;

struct Node \*two = NULL;

struct Node \*three = NULL;

struct Node \*four = NULL;

head = (struct Node \*)malloc(sizeof(struct Node));

one = (struct Node \*)malloc(sizeof(struct Node));

two = (struct Node \*)malloc(sizeof(struct Node));

three = (struct Node \*)malloc(sizeof(struct Node));

four = (struct Node \*)malloc(sizeof(struct Node));

head->data = 0;

head->next = one;

one->data = 1;

one->next = two;

two->data = 2;

two->next = three;

three->data = 3;

three->next = four;

four->data = 4;

four->next = head;

printList(&head);

printf("\n==========Insert======================\n");

insertEnd(&head, 5);

printList(&head);

insertBeg(&head, -1);

printList(&head);

insertMid(&head, 10, 2);

printList(&head);

printf("\n==========Delete======================\n");

deleteBeg(&head);

printList(&head);

deleteMid(&head,3);

printList(&head);

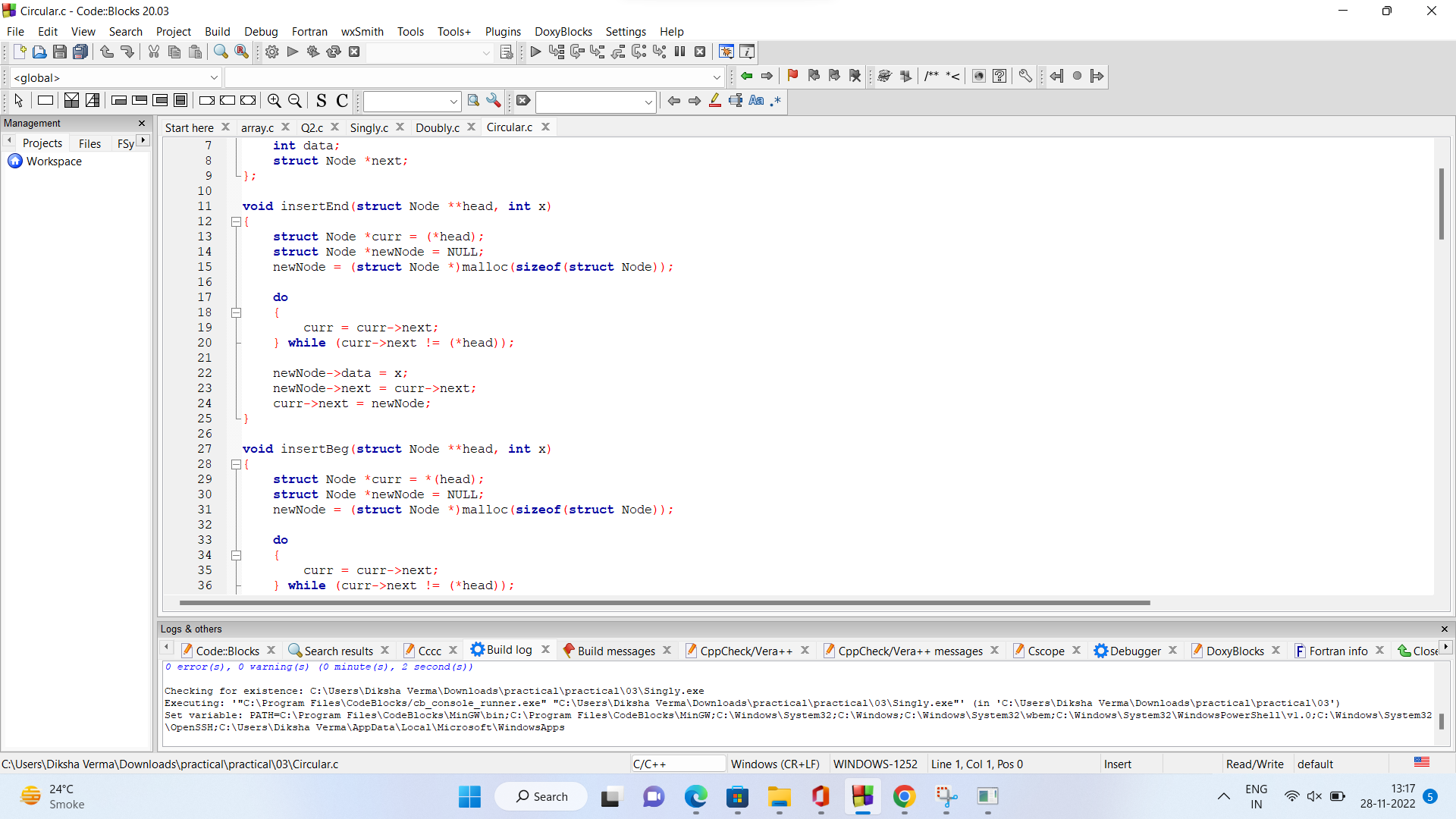
deleteEnd(&head);

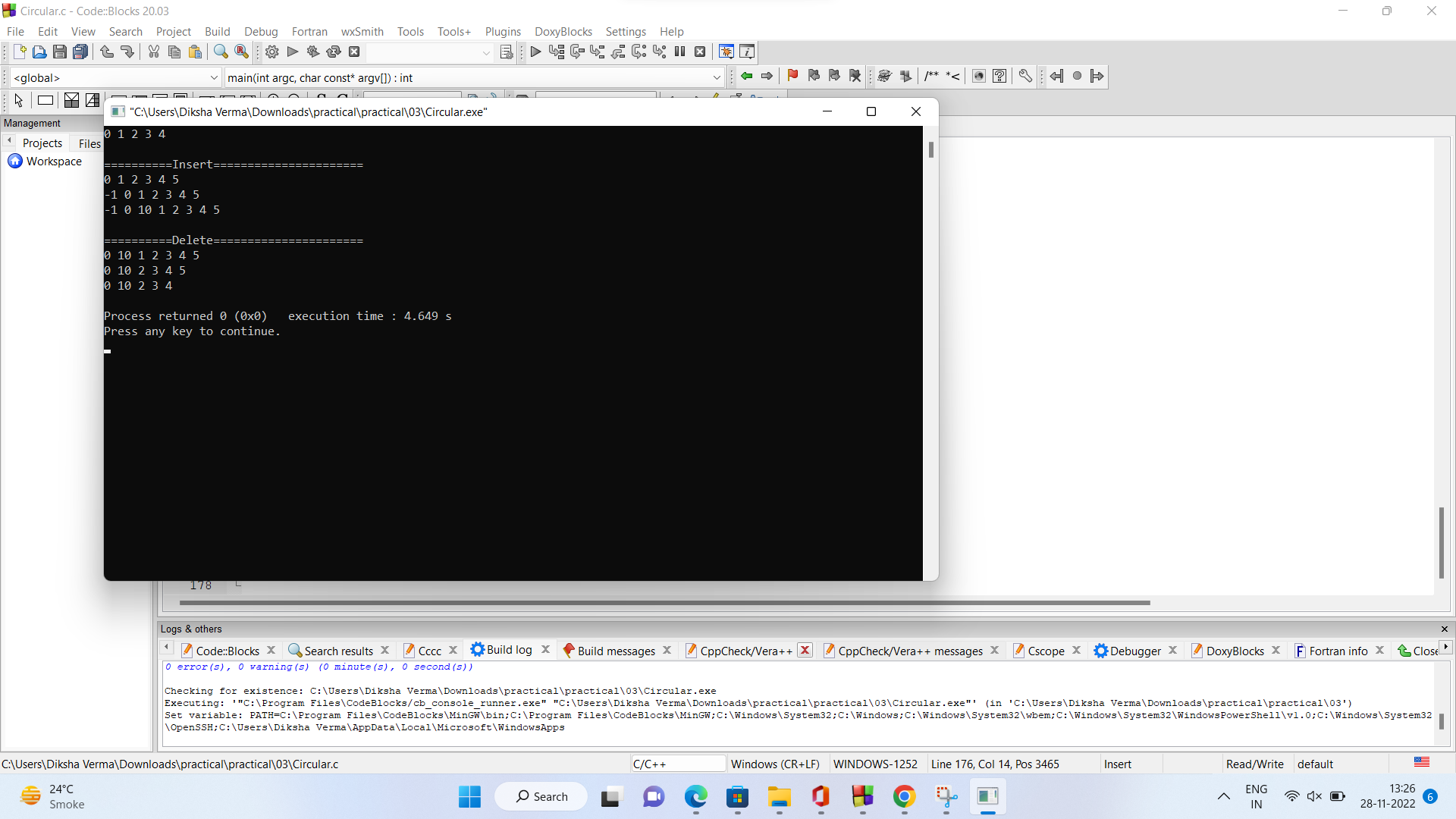
printList(&head);

return 0;

}

Code screenshot





**Practical - 4**

**Q4.Write a menu driven program to implement stack using array .**

Code:

#include<stdio.h>

#include<stdlib.h>

#define SIZE 10

// stack using array//

void push();

void pop();

void display();

int stack\_arr[SIZE] , top ,item, i, count ;

top = -1;

count = 0;

void main(){

int choice = 0;

while(choice!=7){

printf("\n\*\*\*\*\*Main menu\*\*\*\*\* \n");

printf("\nchoose one option from the list");

printf("\n1.push\n2.pop\n3.display\n4.exit\n");

printf("\nEnter your choice");

scanf("%d",&choice);

switch(choice){

case 1:

push();

break;

case 2:

pop();

break;

case 3:

display();

break;

case 4:

exit(0);

break;

default:

printf("plz enter a valid choice");

}

}

}

void push(){

printf("\n\n Enter item to be inserted");

scanf("%d" ,&item);

if(top == SIZE -1){

printf("Overflow");

}

else{

top =top +1;

stack\_arr[top] = item;

}

count++;

}

void pop(){

if(top == -1){

printf("Underflow");

}

else{

item =stack\_arr[top];

top =top -1;

}

count--;

}

void display(){

printf("\n item in satck\n");

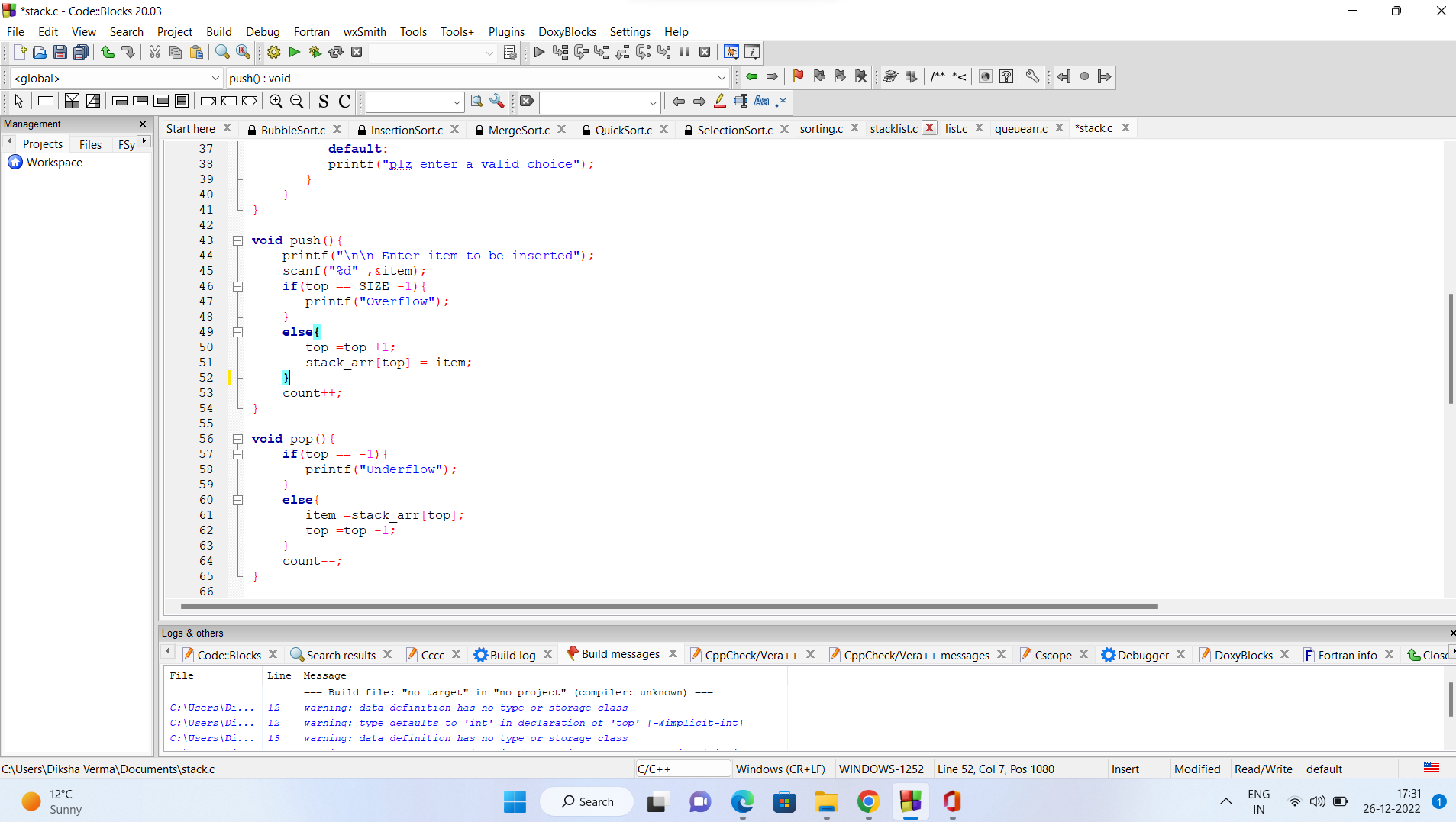
for( i= count-1;i>=0;i--){

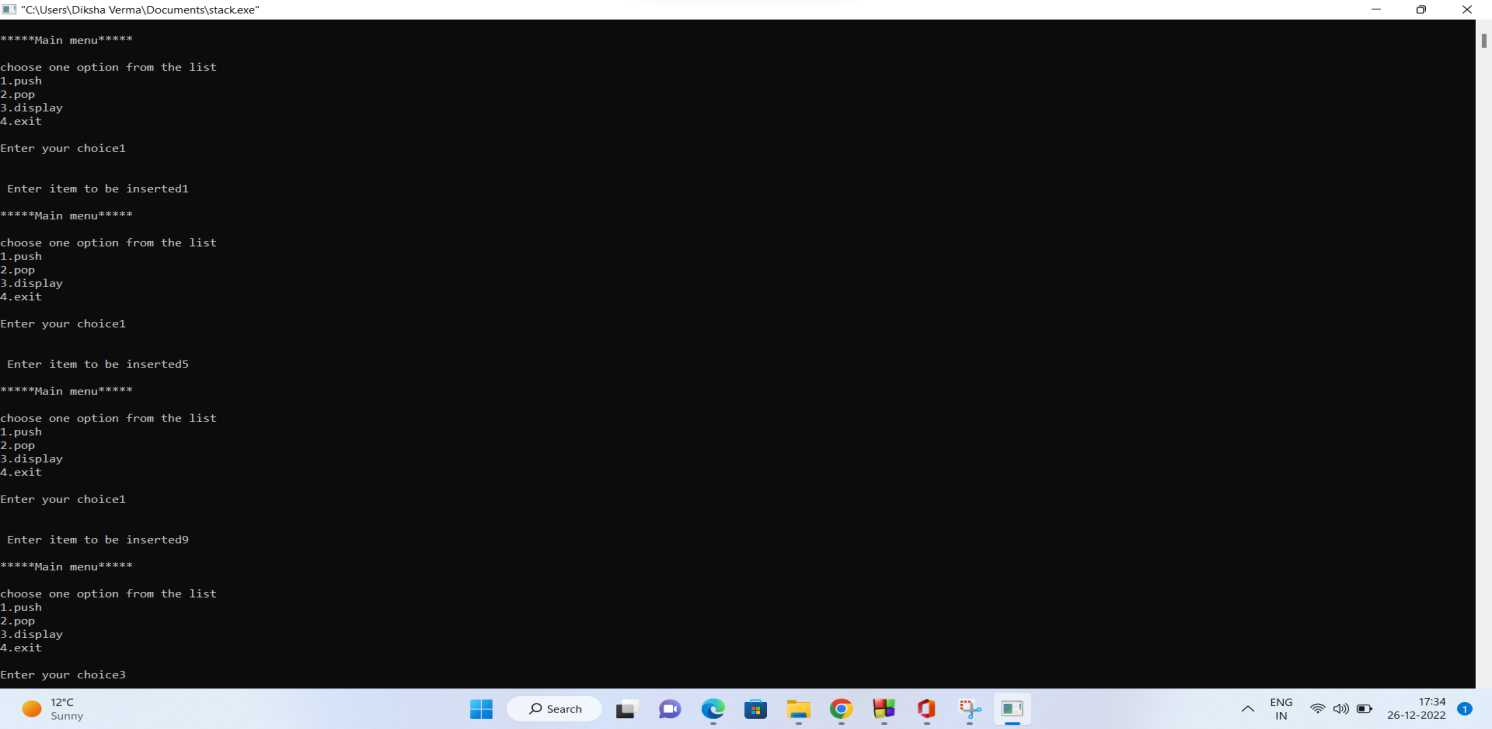
printf("\n%d\n" ,stack\_arr[i]);

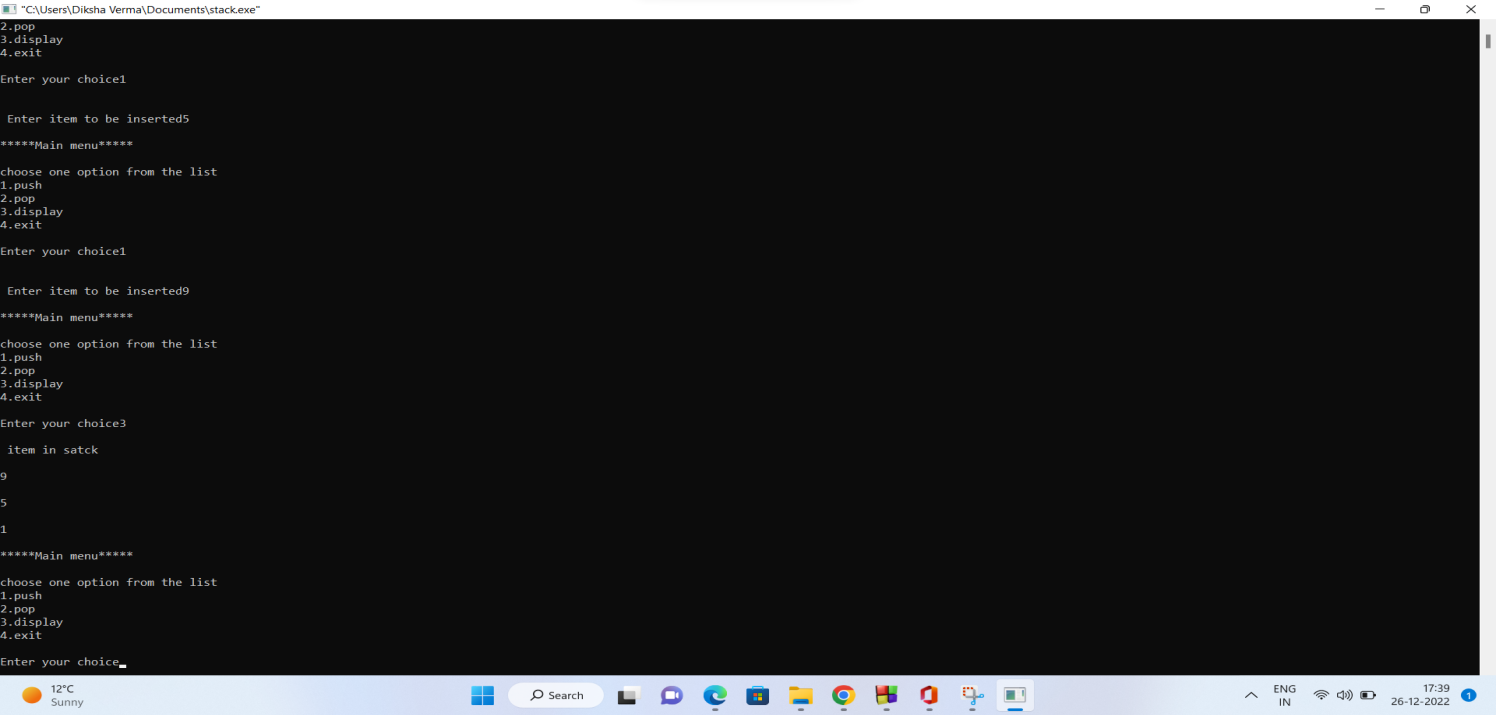
}

}

Output scheenshot







**Practical - 5**

**Q5. Write a menu driven program to implement Queue using using an array.**

Code:

#include<stdio.h>

#include<stdlib.h>

#define MAX 50

void insert();

void del();

void dsiplay();

int queue\_arr[MAX] rear = -1, front = -1;

int main(){

int choice ;

while(1){

printf("\n\*\*\*\*\*Main menu\*\*\*\*\* \n");

printf("\nchoose one option from the list");

printf("\n1.insert\n2.del\n3.display\n4.exit\n");

printf("\nEnter your choice");

scanf("%d",&choice);

switch(choice){

case 1:

insert();

break;

case 2:

del();

break;

case 3:

display();

break;

case 4:

exit(1);

break;

default:

printf("plz enter a valid choice");

}

}

}

void insert(){

int add\_item;

if(rear == MAX -1)

printf("Queue Overflow");

else{

if(front == -1)

front = 0;

printf("Insert the element in Queue :");

scanf("%d" , &add\_item );

rear =rear +1 ;

queue\_arr[rear] = add\_item;

}

}

void del(){

if(front == -1 || front > rear){

printf("Queue if overflow");

return ;

}

else{

printf("Element deleted from queue is :% d\n" , queue\_arr[front]);

front = front +1 ;

}

}

void display(){

if(front == -1)

printf("Queue is empty \n");

else{

printf("Queue is : \n ");

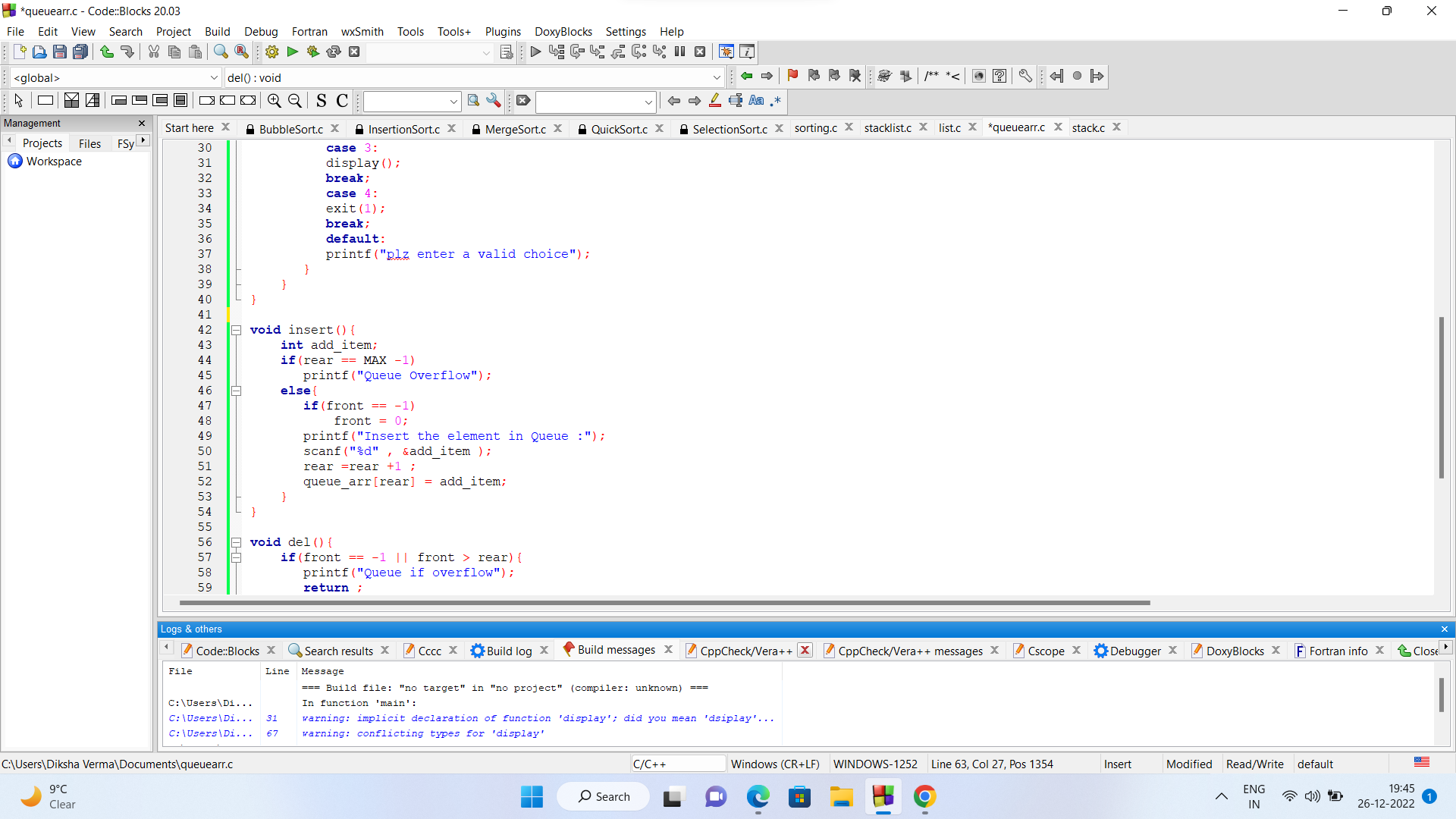
for(int i =front ; i<= rear ; i++)

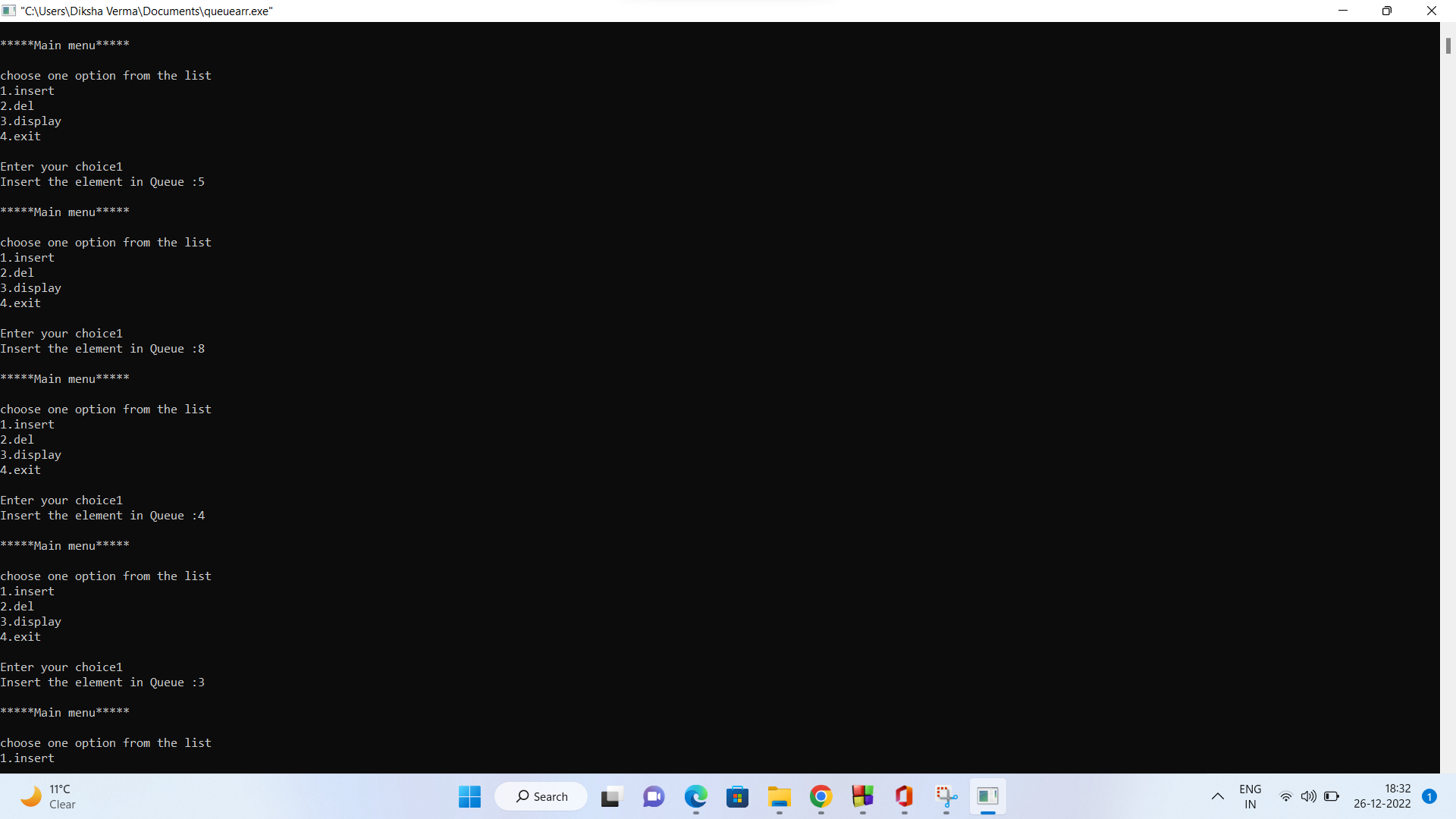
printf("%d" ,queue\_arr[i]);

}

}

Code screenshot





**Practical - 6**

**Q 6.Write a menu driven program to implement circular Queue using array.**

Code:

int size, queue[100],front = -1, rear = -1;

int isFull()

{

if (front == rear + 1 || (front == 0 && rear == size - 1))

return 1;

else

return 0;

}

void enqueue()

{

int item;

if (isFull())

{

printf("Queue overflow\n");

}

else

{

printf("\nInset the element in queue : ");

scanf("%d", &item);

if (front == -1)

{

front = 0;

}

rear = (rear + 1) % size;

queue[rear] = item;

}

}

void dequeue()

{

if (front == -1)

{

printf("Queue Underflow \n");

return;

}

else

{

if (front == rear)

{

front, rear = -1, -1;

}

else

{

front = (front + 1) % size;

}

}

}

void display(){

if (isFull())

{

printf("Queue is Empty \n");

}

else

{

int i;

for (i = front; i < rear; i = (i + 1) % size){

printf("%d ", queue[i]);

}

printf("%d ", queue[i]);

printf("\n");

}

}

int main()

{

int choice;

printf("Enter the size of Circular Queue.. ");

scanf("%d", &size);

while (1)

{

printf("1.Insert element to queue \n");

printf("2.Delete element from queue \n");

printf("3.Display all elements of queue \n");

printf("4.Quit \n");

printf("Enter your choice : ");

scanf("%d", &choice);

switch (choice)

{

case 1:

enqueue();

break;

case 2:

dequeue();

break;

case 3:

display();

break;

case 4:

exit(1);

default:

printf("Wrong choice \n");

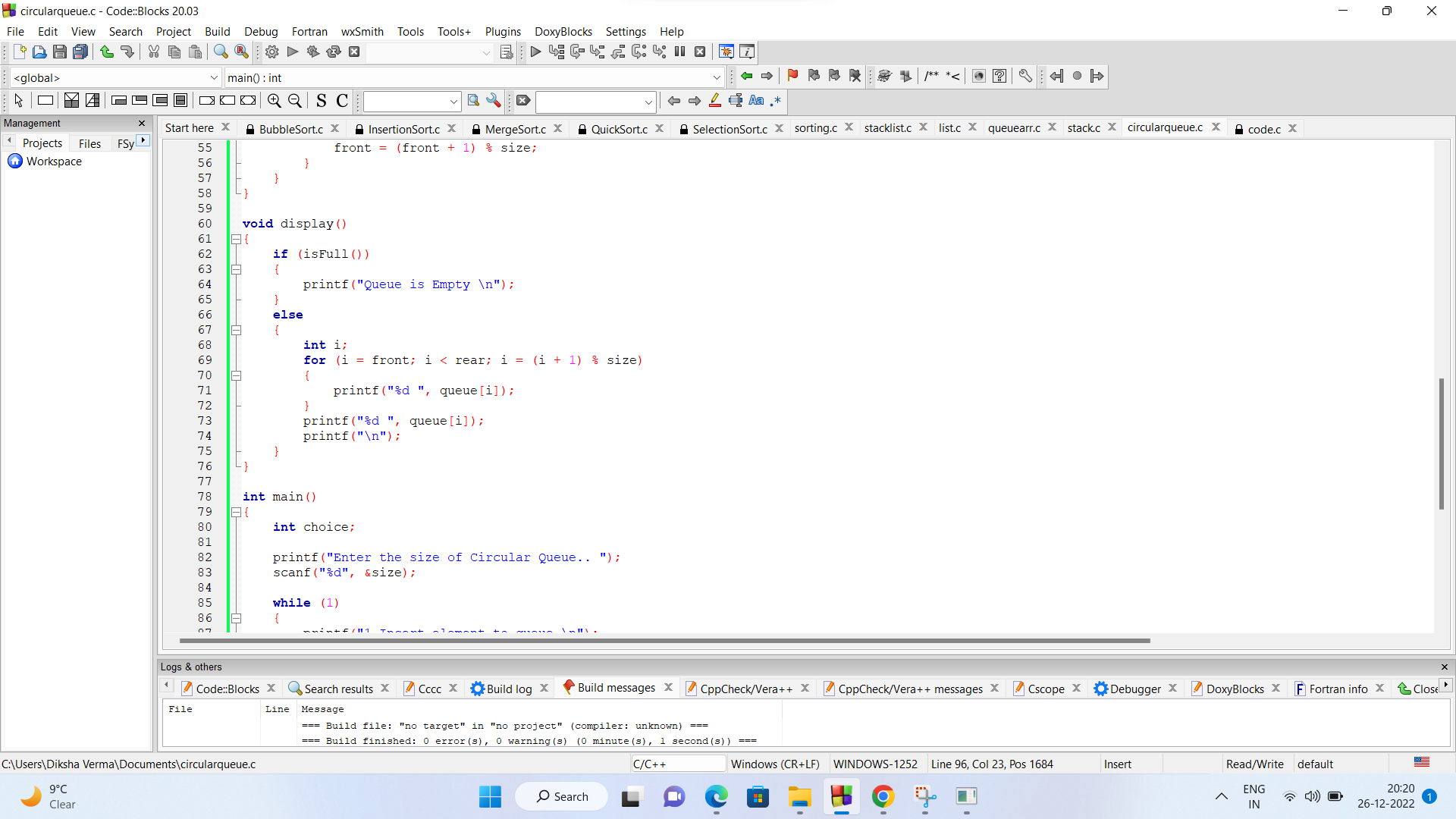
}

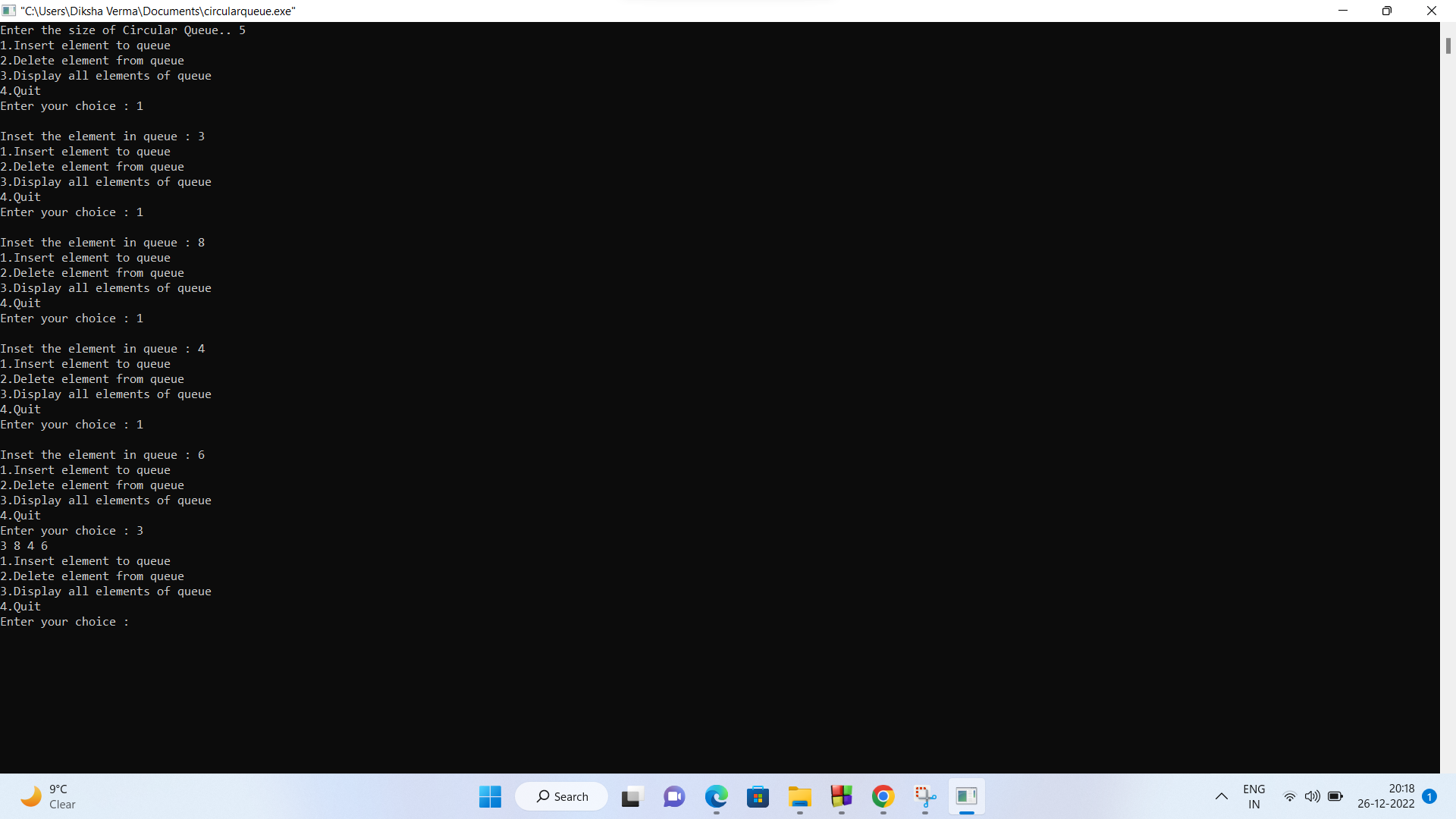
}

return 0;

}

Output screenshot





**Practical - 7**

**Q7.Write a menu driven program to implement stack using linked list.**

Code:

#include <stdlib.h>

#include <stdio.h>

struct Node

{

int data;

struct Node \*next;

};

struct Node \*top = NULL;

void push()

{

int value;

struct Node \*newNode = NULL;

newNode = (struct Node \*)malloc(sizeof(struct Node));

printf("\nEnter element to be inserted to the stack: ");

scanf("%d", &value);

newNode->data = value;

if (top == NULL) {

newNode->next = NULL;

}

else

{

newNode->next = top;

}

top = newNode;

}

void pop()

{

if (top == NULL)

{

printf("\n Underflow\n");

}

else

{

struct Node \*curr = top;

top = top->next;

free(curr);

}

}

void display()

{

if (top == NULL)

{

printf("\n Underflow\n");

}

else

{

struct Node \*temp = top;

while (temp != NULL)

{

printf("%d \n", temp->data);

temp = temp->next;

}

}

}

int main() {

int choice;

printf("Stack using LinkedList... \n ");

while (1)

{

printf("\nOperations performed by Stack");

printf("\n1.Push the element\n2.Pop the element\n3.Show\n4.End");

printf("\n\nEnter the choice: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

push();

break;

case 2:

pop();

break;

case 3:

display();

break;

case 4:

exit(0);

default:

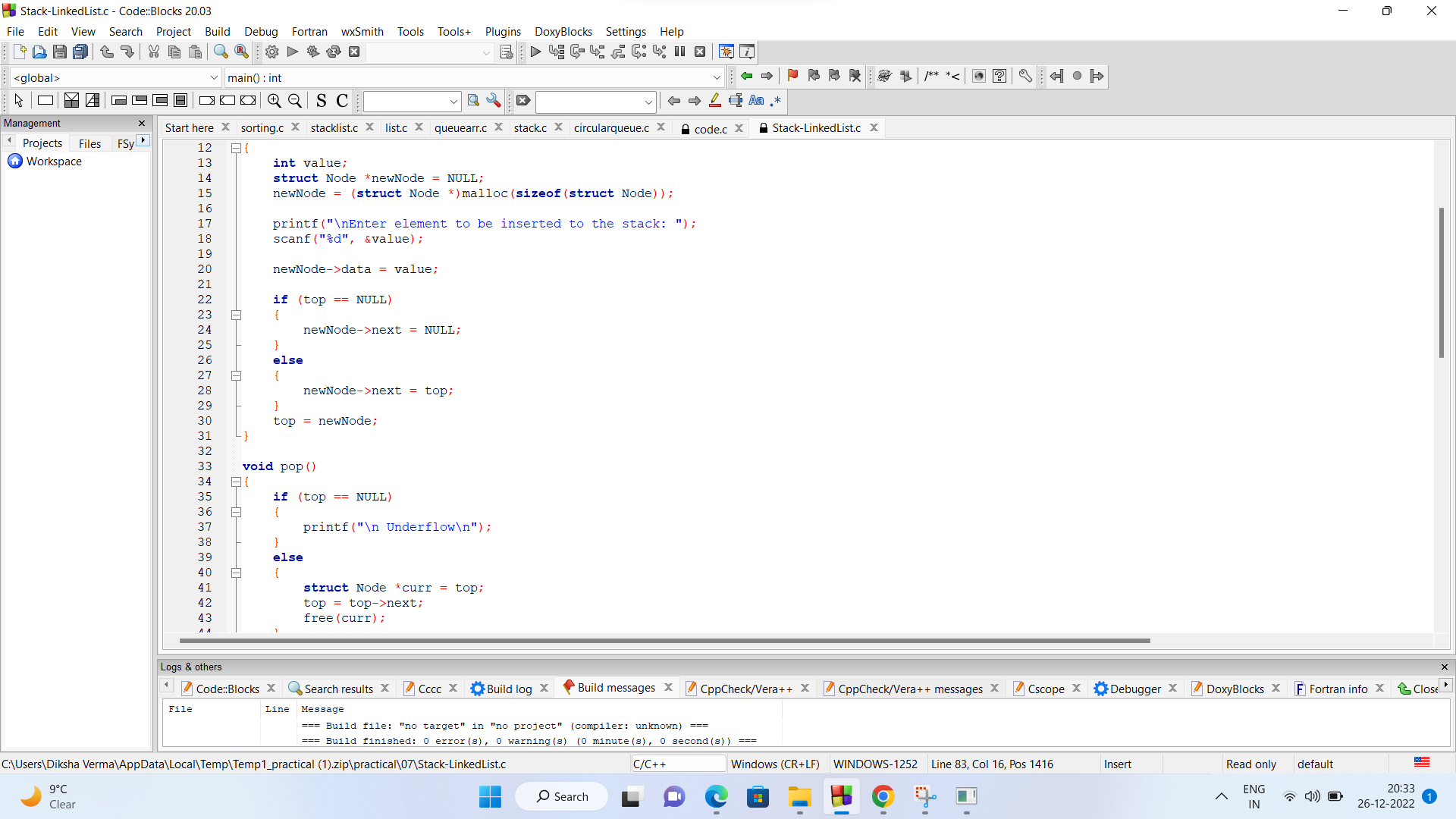
printf("\nInvalid choice!! ");

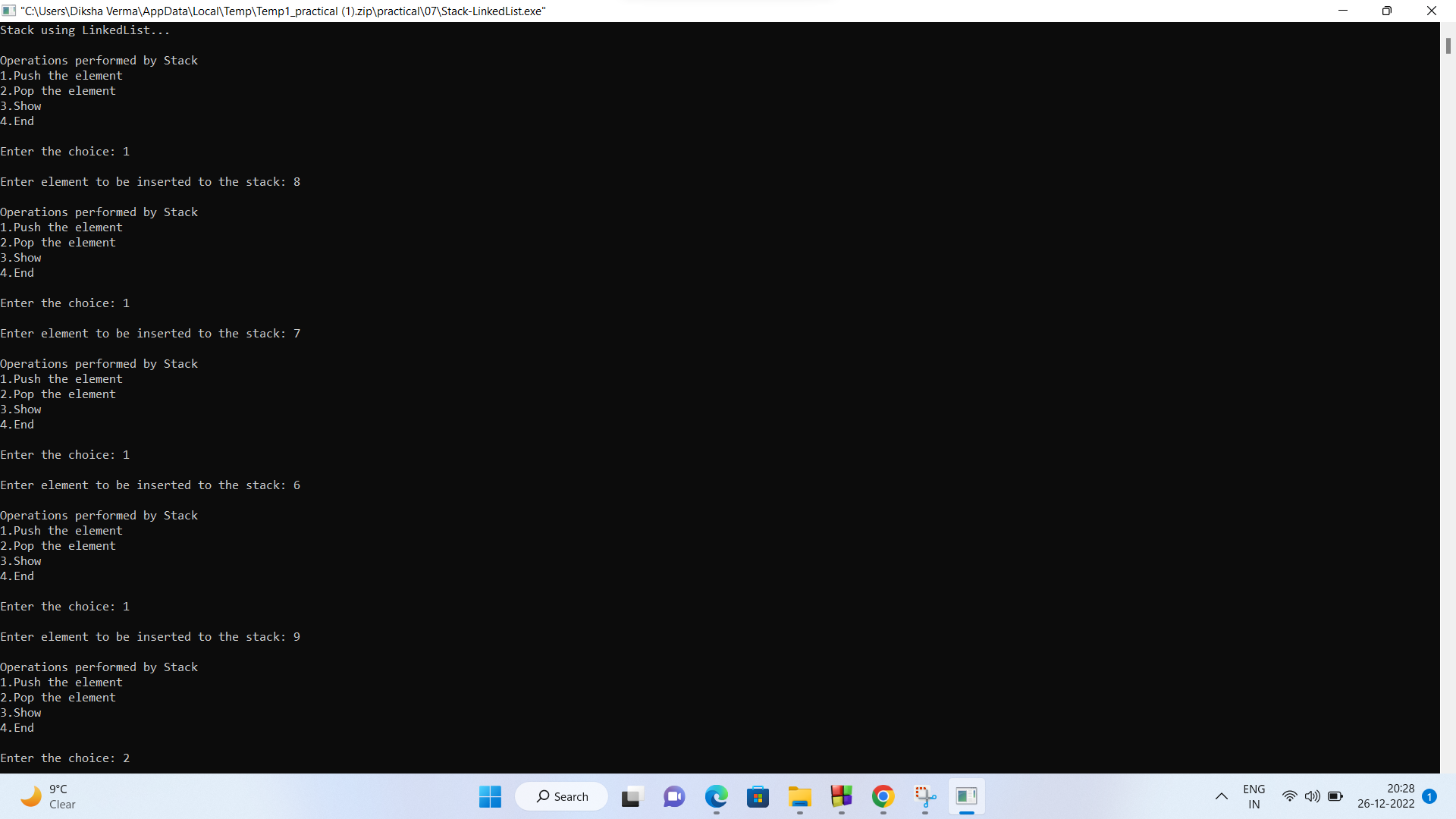
}

}

return 0;

}





**Practical - 8**

**Q8.Write a menu driven program to implement queue using linked list.**

Code:

#include <stdio.h>

#include <stdlib.h>

struct Node

{

int data;

struct Node \*next;

};

struct Node \*head = NULL;

void Enqueue(struct Node \*\*head, int data)

{

struct Node \*newNode = (struct Node \*)malloc(sizeof(struct Node));

newNode->data = data;

newNode->next = NULL;

if (\*head == NULL)

{

\*head = newNode;

return;

}

struct Node \*temp = \*head;

while (temp->next != NULL)

temp = temp->next;

temp->next = newNode;

}

void Dequeue()

{

struct Node \*ptr = head;

head = head->next;

free(ptr);

}

void display(struct Node \*node)

{

while (node != NULL)

{

printf("%d ", node->data);

node = node->next;

}

printf("\n");

}

void main()

{

int choice = 0, item;

printf("Implement Queue using LinkedList \n");

while (choice != 7)

{

printf("\n Choose one of the following list");

printf("\n1.enqueue\n2.dequeue\n3.display\n4.exit\n");

printf("\n enter your choice ");

scanf("%d", &choice);

switch (choice)

{

case 1:

{

printf("enter the item to be inserted ");

scanf("%d", &item);

Enqueue(&head, item);

}

break;

case 2:

Dequeue();

break;

case 3:

display(head);

break;

case 4:

exit(0);

break;

default:

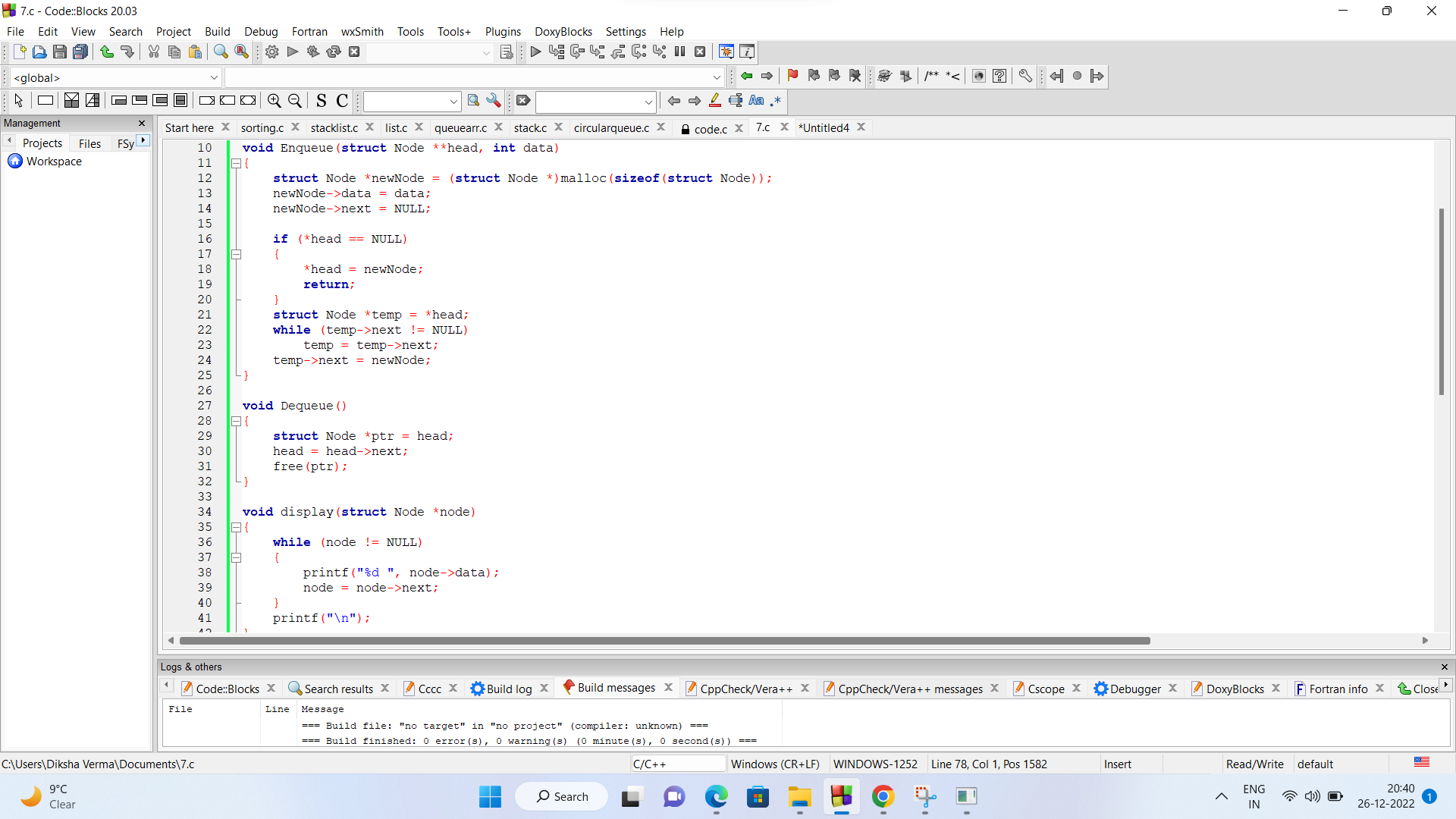
printf("please enter valid choice.... ");

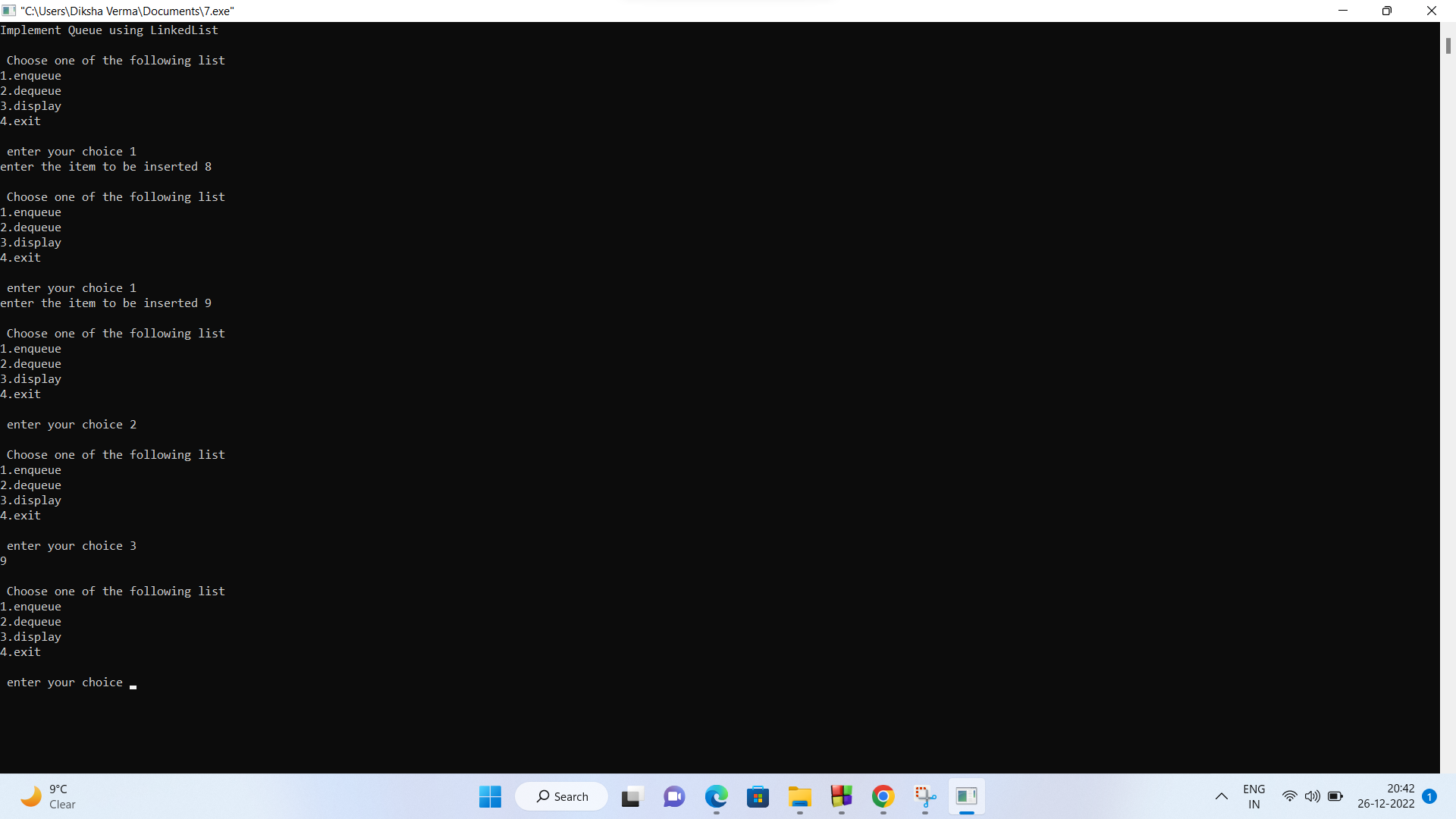
}

}

}

Code screenshot





**Practical - 9**

**Q9.Write a menu driven program to implement a circular queue using a linked list.**

Code:

#include <stdio.h>

#include <stdlib.h>

struct Node

{

int data;

struct Node \*next;

};

struct Node \*head = NULL;

void Enqueue(struct Node \*\*head, int data)

{

struct Node \*newNode = (struct Node \*)malloc(sizeof(struct Node));

newNode->data = data;

if (\*head == NULL)

{

newNode->next = newNode;

\*head = newNode;

return;

}

struct Node \*temp = \*head;

while (temp->next != (\*head))

temp = temp->next;

temp->next = newNode;

newNode->next = (\*head);

}

void Dequeue()

{

if(head->next==head){

free(head);

head=NULL;

return;

}

struct Node \*curr = head;

struct Node \*temp = head;

while (temp->next != curr)

{

temp = temp->next;

}

head = head->next;

temp->next = head;

free(curr);

}

void display(struct Node \*node)

{

if(head==NULL){

printf("Empty Queue...\n");

return;

}

struct Node \*curr = head;

do

{

printf("%d ", curr->data);

curr = curr->next;

} while (curr != (head));

printf("\n");

}

void main()

{

int choice = 0, item;

printf("Implement Queue using LinkedList \n");

while (choice != 7)

{

printf("\n Choose one of the following list");

printf("\n1.enqueue\n2.dequeue\n3.display\n4.exit\n");

printf("\n enter your choice ");

scanf("%d", &choice);

switch (choice)

{

case 1:

{

printf("enter the item to be inserted ");

scanf("%d", &item);

Enqueue(&head, item);

}

break;

case 2:

Dequeue();

break;

case 3:

display(head);

break;

case 4:

exit(0);

break;

default:

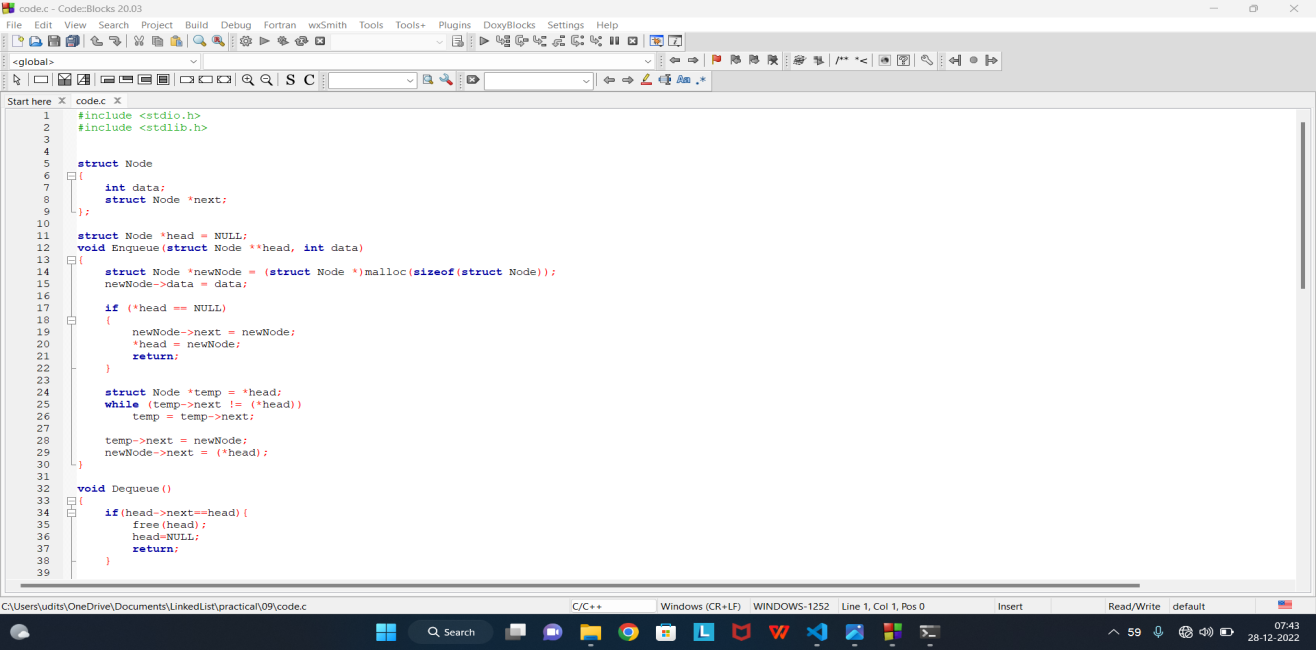
printf("please enter valid choice.... ");

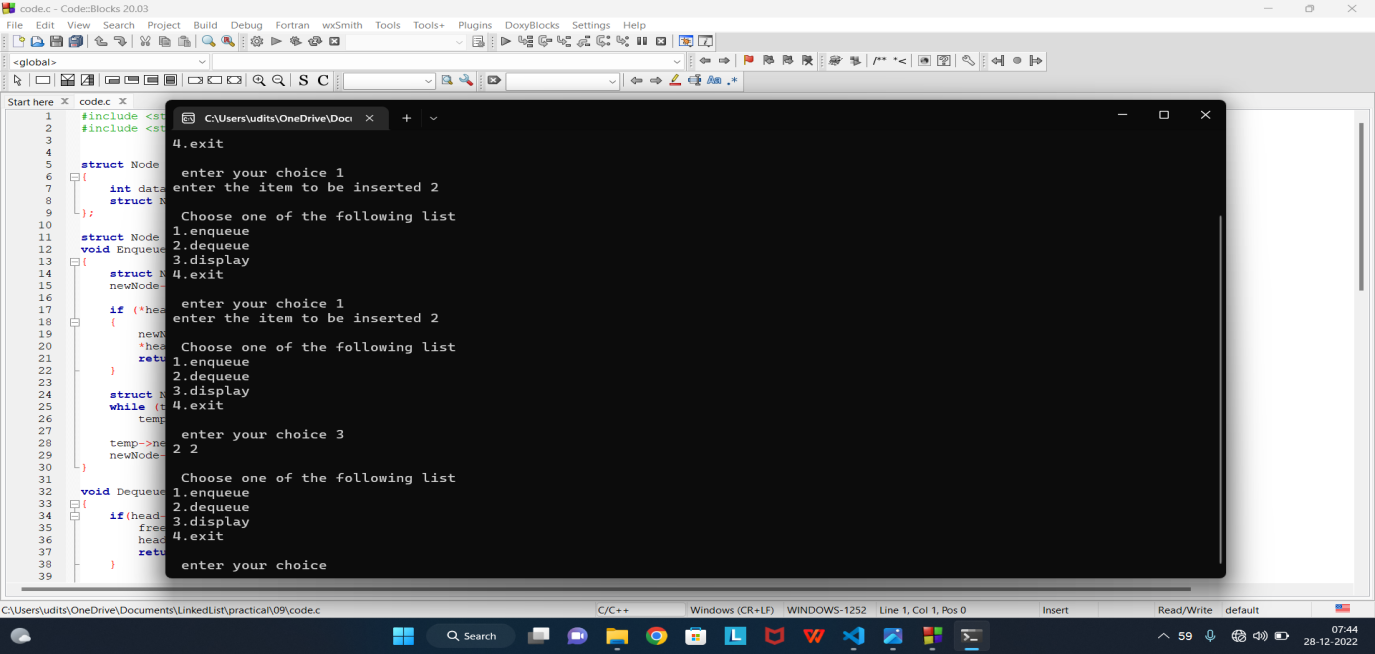
}

}

}

Code Output





**Practical - 10**

**Q10. Write a program to convert infix expression to postfix and prefix conversion**

Code:

#include<stdio.h>

#include<ctype.h>

char stack[100],top = -1;

void push(char x)

{

stack[++top] = x;

}

char pop()

{

if(top == -1)

return -1;

else

return stack[top--];

}

int priority(char x)

{

if(x == '(')

return 0;

if(x == '+' || x == '-')

return 1;

if(x == '\*' || x == '/')

return 2;

return 0;

}

int main()

{

char exp[100];

char \*e, x;

printf("Enter the expression : ");

scanf("%s",exp);

printf("\n");

e = exp;

while(\*e != '\0')

{

if(isalnum(\*e))

printf("%c ",\*e);

else if(\*e == '(')

push(\*e);

else if(\*e == ')')

{

while((x = pop()) != '(')

printf("%c ", x);

}

else

{

while(priority(stack[top]) >= priority(\*e))

printf("%c ",pop());

push(\*e);

}

e++;

}

while(top != -1)

{

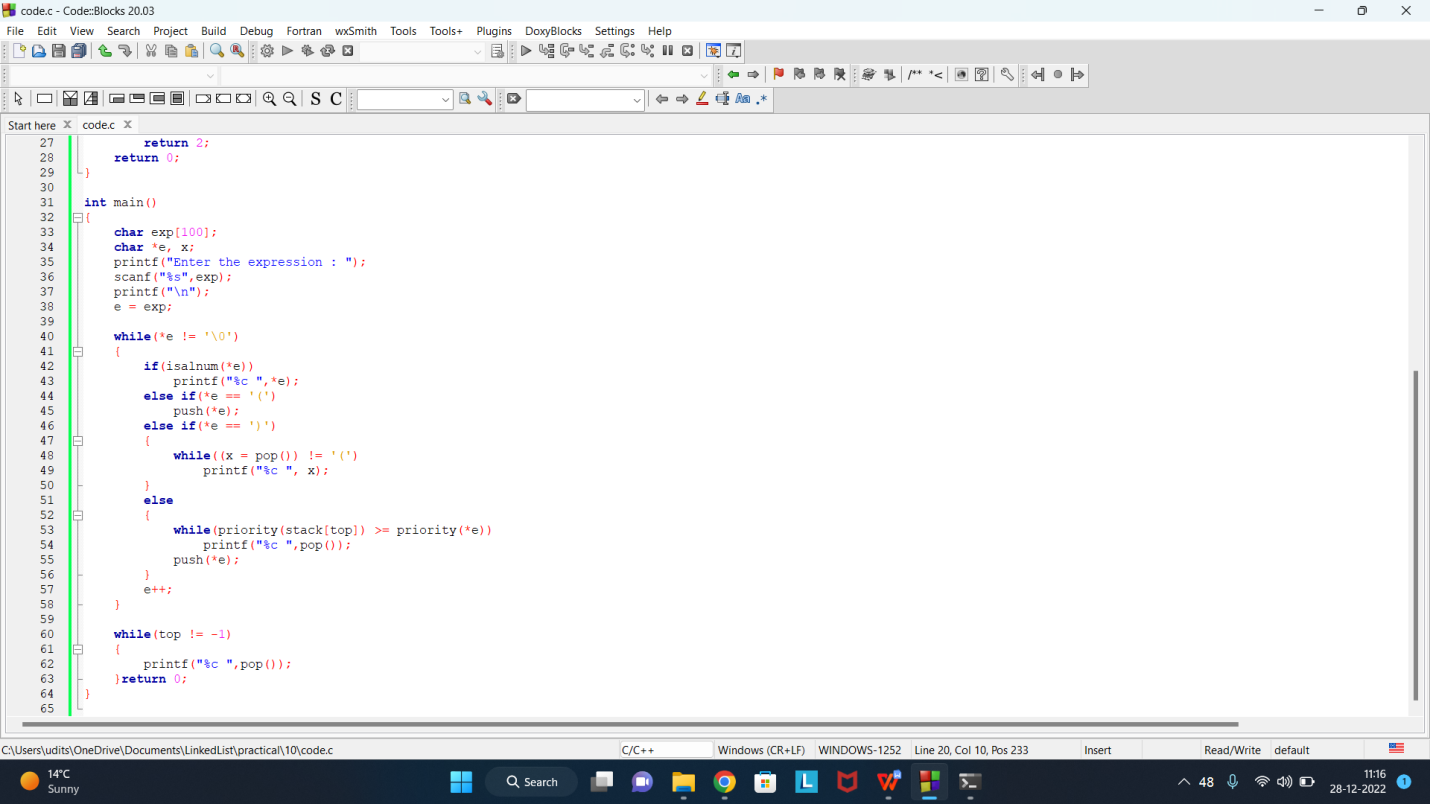
printf("%c ",pop());

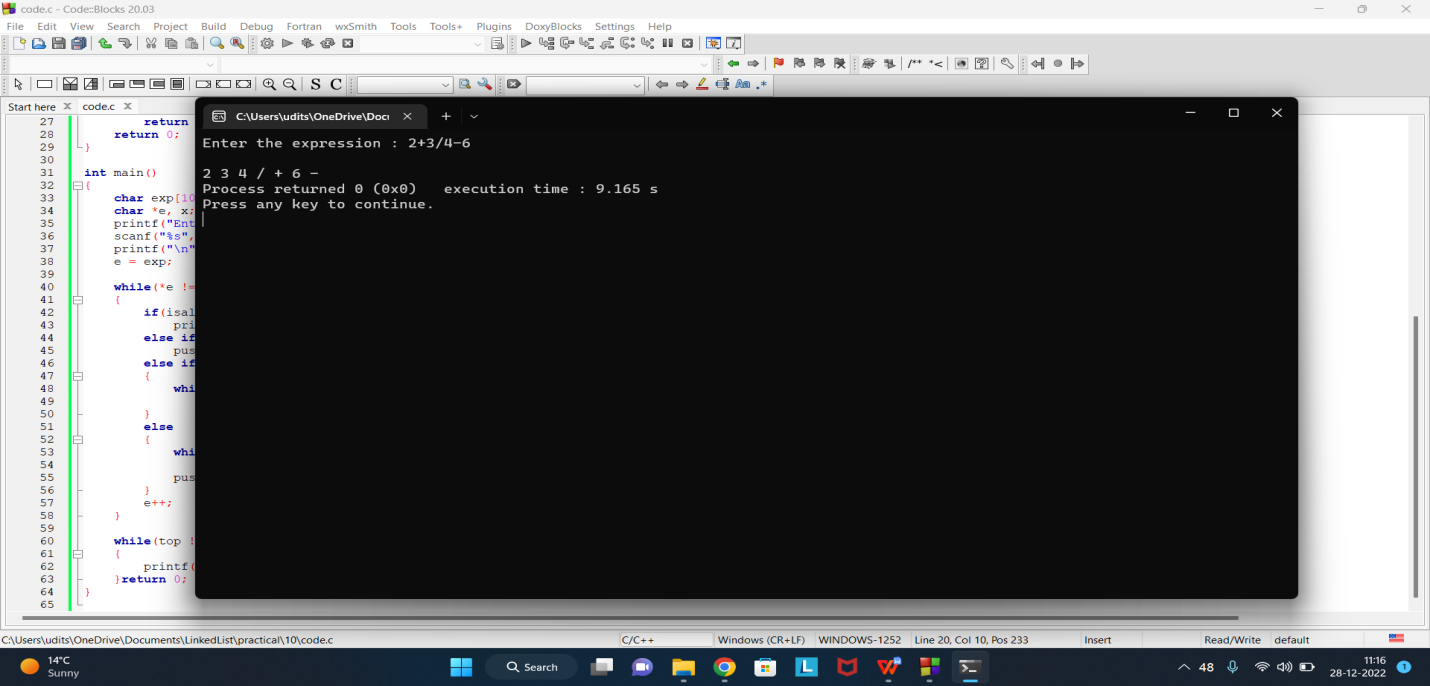
}

return 0;

}

Code screenshot

****

****

**Practical - 11**

**Q11.** **Write a menu-driven program which will accept an array of 10 integer value and sort the with**

1. **Bubble Sort**
2. **Selection Sort**
3. **Insertion Sort**
4. **Merge Sort**
5. **Quick Sort**

Code:

#include <stdlib.h>

#include <stdio.h>

void **swap**(int \*a, int \*b)

{

int t = \*a;

\*a = \*b;

\*b = t;

}

void printArray(int arr[], int n)

{

for (int i = 0; i < n; i++)

{

printf("%d ", arr[i]);

}

}

void **BubbleSort**(int arr[], int n)

{

for (int i = 0; i < n; i++)

{

for (int j = i + 1; j < n; j++)

{

if (arr[i] > arr[j])

{

swap(&arr[i], &arr[j]);

}

}

}

}

void **insertionSort**(int arr[], int n)

{

for (int i = 1; i < n; i++)

{

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key)

{

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

void **SelectionSort**(int arr[], int n)

{

for (int i = 0; i < n; i++)

{

int position = i;

for (int j = i + 1; j < n; j++)

{

if (arr[position] > arr[j])

{

position = j;

}

}

if (position != i)

{

swap(&arr[position], &arr[i]);

}

}

}

void merge(int arr[], int l, int mid, int r)

{

int n1 = (mid - l) + 1;

int n2 = r - mid;

int left[n1], right[n2];

for (int i = 0; i < n1; i++)

{

left[i] = arr[l + i];

}

for (int i = 0; i < n2; i++)

{

right[i] = arr[mid + 1 + i];

}

int i = 0, j = 0, k = l;

while (i < n1 && j < n2)

{

if (left[i] <= right[j])

{

arr[k] = left[i];

i++;

}

else

{

arr[k] = right[j];

j++;

}

k++;

}

while (i < n1)

{

arr[k] = left[i];

i++;

k++;

}

while (j < n2)

{

arr[k] = right[j];

j++;

k++;

}

}

void **mergeSort**(int arr[], int l, int r)

{

if (l < r)

{

int mid = l + (r - l) / 2;

mergeSort(arr, l, mid);

mergeSort(arr, mid + 1, r);

merge(arr, l, mid, r);

}

}

int partition(int array[], int low, int high)

{

int pivot = array[high];

int i = (low - 1);

for (int j = low; j < high; j++)

{

if (array[j] <= pivot)

{

i++;

swap(&array[i], &array[j]);

}

}

swap(&array[i + 1], &array[high]);

return (i + 1);

}

void **quickSort**(int array[], int low, int high)

{

if (low < high)

{

int pi = partition(array, low, high);

quickSort(array, low, pi - 1);

quickSort(array, pi + 1, high);

}

}

int main()

{

int N = 10, choice;

int arr[N];

printf("Enter 10 value in an array: \n");

for (int i = 0; i < N; i++)

{

printf("value %d: ", i + 1);

scanf("%d", &arr[i]);

}

printf("\n Enter your choice\n");

printf("\n 1. Bubble Sort\n 2.Selection Sort\n 3.Insertion Sort\n 4.Merge Sort\n 5. Quick Sort\n ");

scanf("%d", &choice);

switch (choice)

{

case 1:

BubbleSort(arr, N);

break;

case 2:

SelectionSort(arr, N);

break;

case 3:

insertionSort(arr, N);

break;

case 4:

mergeSort(arr, 0, N - 1);

break;

case 5:

quickSort(arr, 0, N - 1);

break;

default:

printf("Invalid Choices....\n");

break;

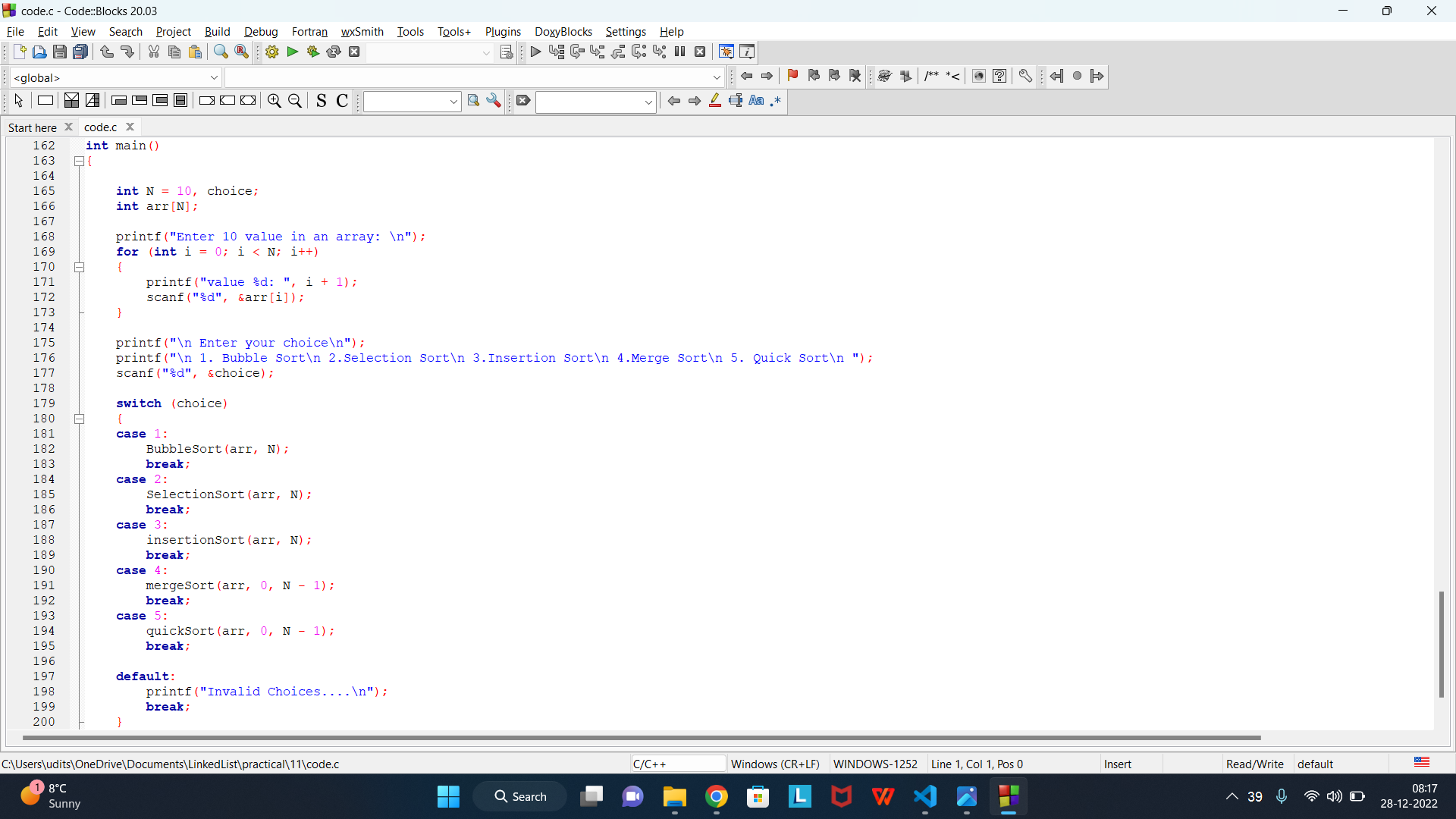
}

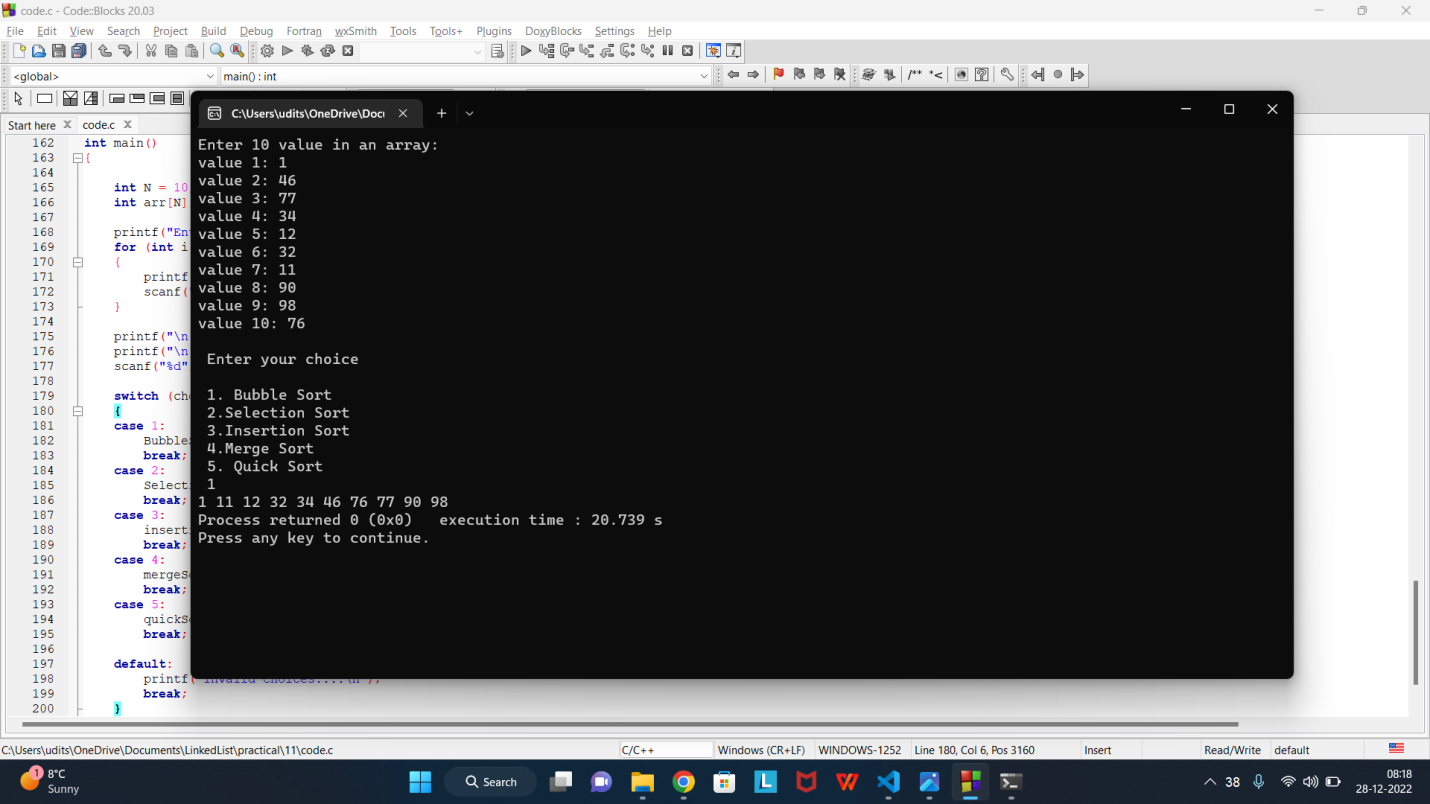
printArray(arr, N);

return 0;

}

Code screenshot





**Practical - 12**

**Q12. Write a function which will accept an array of 20 integer and the value to be searched.**

1. **Linear Search**
2. **Binary Search**

Code:

#include <stdlib.h>

#include <stdio.h>

void insertionSort(int arr[], int n)

{

for (int i = 1; i < n; i++)

{

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key)

{

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

void printArray(int arr[], int n)

{

for (int i = 0; i < n; i++)

printf("%d ", arr[i]);

printf("\n");

}

void BinarySearch(int arr[], int N, int x){

insertionSort(arr,N);

printf("array after sorting...\n");

printArray(arr, N);

int low = 0, high = N, mid;

while (low <= high)

{

mid = ((low + high) / 2);

if (arr[mid] == x)

{

printf("found the value at index %d", mid);

break;

}

else if (arr[mid] > x)

high = mid - 1;

else

low = mid + 1;

}

}

void LinearSearch(int arr[], int N, int x){

insertionSort(arr,N);

printf("array after sorting...\n");

printArray(arr, N);

for (int i = 0; i < N; i++)

if(arr[i]==x){

printf("found the value at index %d ", i);

break;

}

}

int main(){

int N = 20, choice;

int arr[N], x;

printf("Enter 20 integer values for array \n");

for (int i = 0; i < N; i++){

printf("value %d: ", i+1);

scanf("%d", &arr[i]);

}

printf("\nEnter a value to be search: ");

scanf("%d", &x);

printf("\nEnter your choice\n");

printf("1. Linear Search\n2. Binary Search\n");

scanf("%d", &choice);

switch (choice){

case 1:

LinearSearch(arr, N, x);

break;

case 2:

BinarySearch(arr,N,x);

break;

default:

break;

}

return 0;

}

Code screenshot

